Optimising Antibiotic Treatments with Multi-objective Population-based Algorithms

Mila Goranova
Computing Science and Mathematics, University of Stirling
Stirling, United Kingdom
m.g.goranova@stir.ac.uk

Marco A. Contreras-Cruz
Electronics Engineering Department University of Guanajuato, DICIS
Salamanca, Mexico
ma.contrerascruz@ugto.mx

Andrew Hoyle
Computing Science and Mathematics, University of Stirling
Stirling, United Kingdom
andrew.hoyle@stir.ac.uk

Gabriela Ochoa
Computing Science and Mathematics, University of Stirling
Stirling, United Kingdom
gabriela.ochoa@cs.stir.ac.uk

***Abstract*—Antibiotic resistance is one of the major challenges that we are facing today. The frequent overuse of antibiotics is one of the main reasons for the development of resistance. A mathematical model of bacterial population dynamics is used, where drug administration and absorption mechanics are implemented to evaluate the fitness of automatically designed treatments. To maximise the probability of curing the host while minimising the total drug used we have explored treatments with different daily dosages and lengths. Two multi-objective population-based methods, a well-known evolutionary algorithm and a particle swarm optimisation algorithm are tuned and contrasted when solving the posed treatment design problem. The best solutions found by our approach suggest treatments ranging from five to seven days with a high initial dose, followed by lower doses, use lower amounts of the drug than the standard common practice of fixed daily dosages over ten days.**

***Index Terms*—Antibiotics, Treatment Scheduling and Design, Noisy Multi-Objective Optimisation, Stochastic Mathematical Modelling, Pharmacokinetics/Pharmacodynamics Modelling, Evolutionary Algorithms, Particle Swarm Optimisation.**

I. Introduction

Since Penicillin was introduced in 1942, antibiotics have been increasingly prescribed in medicine, agriculture and aquaculture. Their high usage has contributed to the evolution of resistant bacteria strains. These strains are more difficult to treat, or simply cannot be killed by the antibiotics — instead, they survive and multiply [6].

Conventional treatments apply a constant dose for a fixed amount of time — for example, take $60mg$ per day for 10 days. However, medical studies have indicated that shorter treatments can be effective [18]. Other studies have shown that an initial higher dose followed by a lower maintenance dose is beneficial to patients with critical illnesses [12]. Although formulations using bio-inspired algorithms have been proposed for designing cancer chemotherapies [17], [19]–[23], very little work in the literature uses evolutionary algorithms to design tailored antibiotic treatments. A recent study by Cicchese et al. [3] uses a genetic algorithm to design regimens to treat Tuberculosis infections, however, the authors assume that doses are fixed across the treatment, and vary instead the frequency of application of multiple drugs.

This work is looking into optimising antibiotic treatments, which are effective while also short and use the least amount of drug as possible. We extend a mathematical model that simulates the progression of a bacterial infection, first introduced by Paterson et al. [9] where a single-objective evolutionary algorithm was used to design effective treatments. Ochoa et al. also applied a multi-objective evolutionary algorithm in order to automatically design successful antibiotic treatments where constraints and objectives are combined. [16].

The main contributions of this article are as follows:

- To extend the mathematical model used in [9], [16] with a pharmacokinetics/pharmacodynamics (PK/PD) component modelling the antibiotic absorption from the stomach to the blood flow.
- To use a Tau-leaping approach to speed up the simulation time of the stochastic bacteria population model.
- To contrast two population-based algorithms (evolutionary vs. particle swarm optimisation) in the task of optimising a multi-objective formulation of the treatment design problem.

II. Mathematical Model

We followed the mathematical model formulation and parameters detailed in in [9] and [16], where a stochastic model of the progression of a bacterial infection inside a single host, and the effect of antibiotic treatment, is proposed.

When antibiotic treatments are designed, there are two key variables — the daily dosages and the treatment duration. This is modelled as a vector of doses $\underline{x} = (x_1, x_2, ..., x_{10})$, where x_i represents the dosage taken on day i, where $0 \leq x_i \leq 60$ (60 was chosen as the maximum dosage as high levels of antibiotics in the body could be toxic). In this work, we consider a vector of real numbers to encode treatments, $x_i \in \mathbb{R}$, instead of a vector of integer numbers, $x_i \in \mathbb{Z}$ as was the case in [9] and [16]. This allows us to explore more precise prescriptions, which may be relevant considering the recent trend in personalised medicine.

The model follows the steps below:

- Antibiotic dose for the day is taken. The following steps happen until the next dose needs to be taken:
  - Probabilities for bacteria's population death and reproduction are calculated.

- Bacteria population is updated based on the probabilities.
- Time increases by a time-step of 15 minutes.
- New concentration of the antibiotic in the body is calculated.

- Next dose is taken.

The model is detailed in Algorithm 1 where the steps listed above are described further.

A. Parameters and Equations

The equations below show how the reproduction rate and the death rate of the bacteria population (B) are calculated. Table I provides a breakdown with the parameter description and values. All but a, g and p parameter values were chosen by Paterson et al. [9] such that as the concentration of antibiotics increases the death rate will increase as well until it reaches a saturation point. The concentration of antibiotics naturally decays within a host so that was taken into account when choosing the parameters' values.

The parameter values for a and g were chosen so that the maximum concentration of antibiotics in the blood corresponds to the half-life time of the antibiotic - the time required for the concentration of the antibiotic in the body to be reduced by one-half. p was chosen to be 0.8 as the full dosage of antibiotics does not get processed by the metabolism.

$$ReproductionRate_{(bacteria)} = rB\left(1 - \frac{B}{K}\right) \tag{1}$$

$$DeathRate_{(bacteria)} = mB + \frac{(max - min)(\frac{C_{Blood}}{mic})^k}{(\frac{C_{Blood}}{mic})^k - \frac{min}{max}}B \tag{2}$$

TABLE I
LIST OF PARAMETERS AND VALUES. ALL OF THE VALUES EXCEPT FOR a, g AND p ARE TAKEN FROM [9].

Parameter	Description	Value
r	Replication rate of bacteria B	2.7726
K	Carrying capacity	1000
m	Mortality rate of bacteria	0.2
a	Degradation rate of antibiotics in the stomach	1.386
g	Degradation rate of antibiotics in the blood	1.253
p	Proportion of antibiotics that reaches the blood	0.8
max	Max net growth rate in absence of antibiotics	2.5
min	Min net growth rate at high antibiotic levels	-2.1
mic	Min inhibitory concentration (MIC)	16
k	Hill coefficient	4

B. PK/PD

PK/PD modelling is the basis of modern-day pharmacotherapy. Pharmacokinetics describes the drug concentration over time as it courses in the body/host, while the pharmacodynamics observes the effects resulting from the certain concentration of the drug present in the body/host. In other words, pharmacokinetics answers the question 'what the body does to the drug', while pharmacodynamics — 'what the drug does to the body' [10] [13].

In our approach, the PK/PD model is used when calculating the concentration of antibiotics when the daily dose is administered orally. We represent this concentration as $C_{Stomach}$, and its slow decrease during the simulation is calculated by the following equation:

$$\frac{dC_{Stomach}}{dt} = -aC_{Stomach} \tag{3}$$

However, for the antibiotics to be effective, they need to reach the bloodstream where they can fight the bacterial infection. The equation below calculates the level of antibiotics in the blood during the day:

$$\frac{dC_{Blood}}{dt} = paC_{Stomach} - gC_{Blood} \tag{4}$$

As it could be observed from Equation 2 the death rate of the bacteria is correlated with the concentration of the antibiotic — the higher the concentration, the higher the death rate is.

C. Tau-leaping

To save computation power and produce faster results a Tau-leaping approach is taken in this work. Tau-leaping is an approximate method for the simulation of a stochastic system based on the Gillespie algorithm [8]. In Algorithm 1 we use a time-step of 15 minutes for the approximation as this value gives a good balance for speeding up the process without losing too much accuracy. For each time-step, an approximation is calculated for how much the bacteria population has decreased ($DeathRate_{(bacteria)}$) and increased ($ReproductionRate_{(bacteria)}$). Then the bacteria population is updated by taking the $bacteria$ from the previous time-step adding the $ReproductionRate_{(bacteria)}$ and subtracting the $DeathRate_{(bacteria)}$, as shown below

$$B_{(t+\tau)} = B_{(t)} + Poisson(\tau ReproductionRate_{(bacteria)}) - Poisson(\tau DeathRate_{(bacteria)}) \tag{5}$$

This mathematical formula is corresponds to lines 15 to 17 in Algorithm 1 where $\tau = timestep$.

As this is a stochastic model, for each dosage regimen, we run the model 500 times, and record the number of these runs where the antibiotic successfully cleared the bacterial infection by the end of the treatment period.

D. Objectives

We consider a bi-objective formulation of the problem of optimising antibiotic treatments. Specifically, the two objective functions to be minimised are:

- The percentage of runs of the mathematical model where the bacteria survives the treatment — *failure rate* f_{fr}.
- The total amount of antibiotics used, as measured by the sum of the entries in the dosage vector — *total antibiotic* f_{ta}.

Algorithm 1 Outline of the simulation model

$treatment$ = $\{x_1, x_2, \ldots x_n\}$
$bacteria$ = 1000
$concentration_{blood}$ = 0
$concentration_{stomach}$ = 0
$time$ = 0
$timestep$ = 15 minutes
end_of_day = 1440 minutes
for day 1 until the last day of treatment **do**
  $concentration_{stomach}$ = $concentration_{stomach}$ + $treatment_{x_{day}}$
  while $time \leq end\_of\_day$ or $bacteria > 0$ **do**
    $ReproductionRate_{(bacteria)}$ is calculated using Equation 1.
    $DeathRate_{(bacteria)}$ is calculates using Equation 2.
    $bacteria\_increase$ = $Poisson(timestep \times ReproductionRate_{(bacteria)})$
    $bacteria\_decrease$ = $Poisson(timestep \times DeathRate_{(bacteria)})$
    $bacteria$ = $bacteria$ + $bacteria\_increase$ - $bacteria\_decrease$
    $time = time + timestep$
    $concentration_{stomach}$ is calculated by solving Equation 3.
    $concentration_{blood}$ is calculated by solving Equations 3 and 4.
  end while
end for

III. Computational Methods

A. Implementation of the objectives

The two objectives described in Section II-D are *failure rate* f_{fr} and total amount of antibiotics taken f_{ta}. The failure rate is estimated by running the mathematical model with a fixed number of simulations and returning the number of runs in which the bacteria population was not eliminated. For example the treatment vector x = (44.92, 60, 53.17, 39.25, 60, 1.70) has a f_{ta} = 259.04mg and f_{fr} = 0.0391 which could be read as 3.91%.

The failure rate is determined by the stochastic model and due to the random elements in the approach, one treatment could have different outcomes per run — failure (the bacteria population does not get eradicated) or success (bacteria population is eradicated). The model is run several times for the same treatment and the failure rate is determined by how many times the outcome was a fail. The number of model runs had to be carefully selected to minimise the noise, but also minimise the number needed to reduce the computational effort. After preliminary investigations, we have chosen to use 500 model simulations for estimating the failure rate for the candidate treatment solutions.

B. Population-based algorithms

We considered two population-based algorithms as described below. Our implementation used the Python library JmetalPy [1].

1) Non-dominated Sorting Genetic Algorithm II (NSGA-II): NSGA-II uses non-dominated sorting of individuals in the population, with a crowding distance penalty applied to individuals to maintain a diverse Pareto-front [5]. It is one of the best-known and widely used algorithms and was previously used for optimising antibiotic treatments in [16].

2) Speed-constrained Multi-objective Particle Swarm Optimisation Algorithm (SMPSO): SMPSO [14], [15] is a multi-objective particle swarm optimisation algorithm that uses a strategy to limit the velocity of the particles. This strategy allows for producing new effective particle positions when the velocity becomes too high. It also includes polynomial mutation and an external archive to store the non-dominated solutions found during the search. SMPSO produced remarkable results when compared to NSGA-II on a number of standard benchmark functions.

C. Hypervolume Indicator

To compare the two algorithms' performance, we use the hypervolume indicator (also known as Lebesgue measure or S metric) as a measure. The hypervolume indicator is a set measure to evaluate the performance of multi-objective algorithms using a reference point. In our case, this is the reference point with maximum total antibiotics and the reference point with maximum failure rate as we are minimising our objective functions. These reference points are the vector of the maximum 10 days period with the maximum dosage of 60: $\underline{x}$ = $[60, 60, 60, 60, 60, 60, 60, 60, 60, 60]$ with the maximum possible f_{ta} = $600mg$ and the maximum possible failure rate f_{fr} = 100%. The reference point to the Pareto-front space is measured to produce a single number — the hypervolume indicator. As our two objective functions need to be minimised, the higher the hypervolume indicator the better the solution is. The implementation of the calculation for the hypervolume used in this work is the one from Fonseca et al. [7].

D. Parameter Tuning

We used the same configuration effort to tune the multi-objective optimisation algorithms in the design of antibiotic

treatments. We applied an automatic configurator to find the highest performing configurations of the algorithms to avoid a bias in the performance of the techniques and to develop a fair comparison.

We selected the software package irace [11]. This software has been applied to a wide variety of configuration tasks, which include not only tuning the numeric parameters of multi-objective optimisations but also designing automatically new multi-objective optimisation algorithms [2].

In our study, we only configured the numerical parameters of the algorithms by using the iterated F-race implemented in irace.

In iterative F-races, configurations are sampled according to a particular distribution (that evolves with time), and the configurations are evaluated with a racing method to find the optimal configuration. The racing consists of evaluating the performance of the configurations with a sequence of training instances and developing the Friedman test (a statistical test that is used to detect differences in variables across multiple test attempts [4]) to remove statistically worse configurations. This process is repeated until a stopping criterion is met — for example, a maximum number of executions of the algorithm.

During the configuration, the NSGA-II algorithm used the SBX crossover, the polynomial mutation, and the binary tournament selection with the ranking and crowding distance comparator. The SMPSO algorithm used the polynomial mutation operator, and the leader replacement based on crowding distance archive. For the configuration of the numerical parameters, we used the iterated F-race with 120 executions of the algorithm. Each run of the algorithm executed a maximum number of 5000 function evaluations, with 500 runs of the mathematical model. The real numbers consider four decimal places in the configuration. Tables II and III describe the numerical parameters of both algorithms and the results of the tuning procedure.

TABLE II
NSGA-II DEFAULT AND TUNED PARAMETER VALUES.

Parameter	Domain	Default	Tuned
Population size	(30,100)	100	62
Offspring population	(30,100)	100	70
Mutation probability	(0.0,1.0)	0.1	0.0971
Mutation distribution index	(5.0, 400.0)	20	306.2005
Crossover probability	(0.0, 1.0)	1	0.5084
Crossover distribution index	(5.0, 400.0)	20	128.7306

TABLE III
SMPSO DEFAULT AND TUNED PARAMETER VALUES.

Parameter	Domain	Default	Tuned
Swarm size	(30,100)	100	59
Mutation probability	(0.0,1.0)	0.1	0.2821
Mutation distribution index	(5.0, 400.0)	20	307.3271
Size of the archive	(30, 100)	100	56

IV. RESULTS

A. Hypervolume Comparison of Algorithms

Figure 1 shows the hypervolume results of 30 runs for each of the multi-objective algorithms — NSGA-II with default parameters, NSGA-II with tuned parameters, SMPSO with default parameters and SMPSO with tuned parameters.

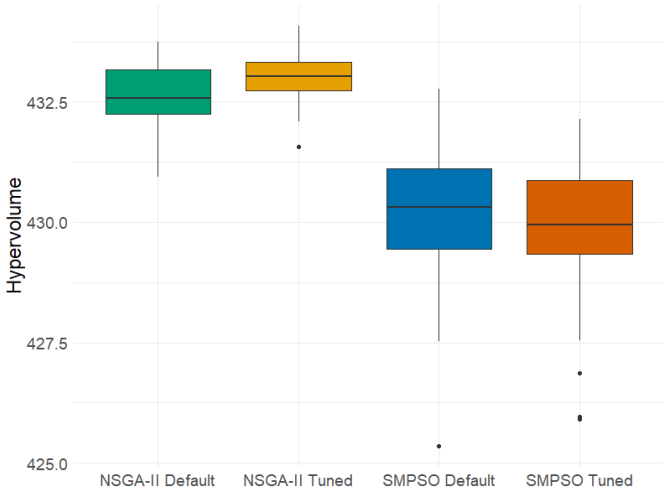

Fig. 1. Distribution of the hyper-volume metric for the two algorithms and parameter settings.

1) NSGA-II and SMPSO Performance: From Figure 1 we can clearly observe that the solutions produced by NSGA-II performed better in regards to the hypervolume compared to SMPSO. The median for the hypervolume for all 60 runs (both tuned and with default parameters) of NSGA-II is 432.8654 and the one for SMPSO is 429.9859. Using the Welch Two Sample t-test, which tests whether the means of two populations are equal, gives a p-value< 0.0001 showing that the performance of the two algorithms is significantly different. The Wilcoxon Signed Rank Test, which tests whether two samples follow the same distribution, and the Kolmogorov-Smirnov Test, which tests whether the mean of two samples are equal, both have p-values< 0.0001, proving the same conclusion that NSGA-II outperforms SMPSO.

2) NSGA-II - tuned and with default parameters: It appears that the tuned NSGA-II performs slightly better than the default parameters NSGA-II in terms of hypervolume results. The median of the tuned NSGA-II is 433.0052 and the default parameter NSGA-II's median hypervolume is 432.5983. The best Pareto-front's hypervolume indicator is significantly higher as well, the outlier is still better than the lowest-performing Pareto-front by the default parameter NSGA-II. This result was expected as the p-value = 0.01258 from Welch Two Sample t-test. This proves the hypothesis that the tuned NSGA-II performs better than the default parameter one. Running the Wilcoxon Signed Rank Test the $p - value$ = 0.01317 and Kolmogorov-Smirnov Test the p-value = 0.03458. The p-values from the two tests are still less than 0.05 so this supports the hypothesis as well.

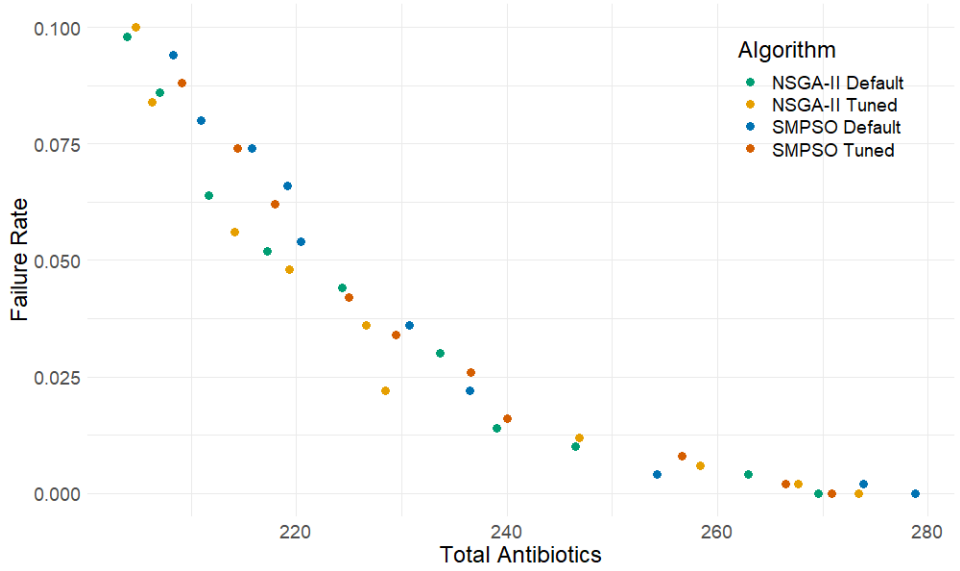

Fig. 2. All solutions with Failure Rate $\leq 10\%$ (0.10) and Total Antibiotics $\leq 300mg$ from 120 runs of the full multi-objective model using the population-based algorithms NSGA-II and SMPSO. The figure combines runs from both default parameters and tuned parameter runs of the two algorithms.

3) SMPSO - tuned and with default parameters: In the case of SMPSO, the tuned algorithm does not perform significantly better than the default parameter - in fact, it initially appears that the default parameters outperforms the tuned. The median of the tuned SMPSO is 429.75 and the median of the default parameter SMPSO is 430.09. However, the Welch Two Sample t-test, with p-value = 0.4189, shows that there is no significant difference, positive or negative, between the tuned or default parameters for SMPSO. The Wilcoxon Signed Rank Test (p-value = 0.7091) and Kolmogorov-Smirnov Test (p-value = 0.808) also show there is no evidence of a difference in performance between the two. This indicates that maybe a bigger budget for the parameter tuning is needed when tuning a PSO-based algorithm.

B. Partial Pareto-front of best results for NSGA-II and SMPSO

Figure 2 shows the 10 best solutions obtained from the 30 runs of each of the algorithms (NSGA-II with default parameters and with tuned, SMPSO with default parameters). Here, we define 'best' as those solutions with the lowest **Failure Rate (%)**. The NSGA-II with tuned parameters points coloured in yellow is visibly producing the most non-dominated solutions followed by NSGA-II with default parameters (coloured in green). This again adds evidence for the hypothesis suggested that NSGA-II outperforms SMPSO.

C. Best Treatments

Table IV shows the top 10 overall best solutions. The solutions are ordered by Re-Evaluated Failure Rate from lowest to highest. Here, we refer to best as the solutions with lowest Failure Rate and lowest Total Antibiotics for that rate.

When designed, the treatments could be of lengths between 3 to 10 days and the upper bound for a single dosage is set to 60mg. In cases where there is a dosage below $1mg$, it has been rounded down to $0mg$ as it makes little difference to the failure rate; the failure rate was then re-evaluated (as discussed below). For example the treatment $\underline{x}$ = (59.892, 41.879, 38.623, 56.518, 25, 46.632, 0.558, 0.48, 0, 0.004), where the initial length of the treatment is 10 days and the last treatment is so close to 0mg that it has been rounded down. After rounding up the dosages to the second decimal this treatment is listed in the table as $\underline{x}$ = (59.89, 41.88, 38.62, 56.52, 25, 46.63) where the final length is 6 days.

All of the solution vectors listed in Table IV were re-evaluated using the mathematical model with 10,000 runs. It was expected for some differences in the Failure Rate to be present due to the noise produced by the stochastic characteristic of the mathematical model. During the iterations of the two multi-objective algorithms, the mathematical model was run 500 times which could contribute further to the noise in the failure rate evaluation.

In Table IV, the top 10 of the solutions have a difference over 5% between the failure rate and the re-evaluated failure rate which is a lot higher than expected but could be explained by the stochastic characteristic of the mathematical model.

Another observation about the solutions listed in Table IV is that only two out of the ten were generated using the SMPSO algorithm (one solution with tuned parameters and one with default parameters). Five out of the ten solutions were generated with NSGA-II with default parameters and three with tuned parameters. This is another indication of the better performance of NSGA-II over SMPSO.

Figure 3 shows the top 5 treatments from Table IV, which have produced the lowest failure rate after the re-evaluation. Each of the treatments is represented by a barplot where each of the bars is the amount of drug for each day of treatment. The dose for the day could be seen at the top of each of the

TABLE IV

THE BEST 10 TREATMENTS OBTAINED BY OUR APPROACH. THEY ALL HAVE FAILURE RATE BETWEEN 0 AND 0.10. THE TREATMENTS ARE ORDERED BY *Re-eval. Failure Rate* STARTING FROM THE LOWEST. THE LENGTH OF THE TREATMENT, AS WELL AS THE ALGORITHM THAT HAS PRODUCED THE RESULT, ARE PROVIDED.

Treatment Vector	Length (days)	Total Antibiotics	Failure Rate (%)	Re-eval. Failure Rate (%)	Algorithm
56.46, 46.70, 51.21, 41.14, 31.05, 46.82	6	273.38	0	2.17	NSGA-II Tuned
59.89, 41.88, 38.62, 56.52, 25, 46.63	6	268.54	0	2.78	NSGA-II Default
44.92, 60, 53.17, 39.25, 60, 1.70	6	259.04	0.4	3.91	SMPSO Default
58.01, 45.78, 55.60, 40.94, 51.85	5	252.18	0.8	4	NSGA-II Tuned
59.95, 44.53, 52.70, 36, 55.05, 0, 7	7	255.23	0.6	4.6	NSGA-II Default
58.75, 47, 57.36, 37.71, 37.71	5	238.53	1.4	5.57	NSGA-II Default
53, 53, 54.01, 29, 54.47	5	243.48	1.2	6.68	NSGA-II Tuned
57.14, 50, 45.66, 49.93, 20.67, 13.56	6	236.96	0	6.84	SMPSO Tuned
56.18, 44.96, 45.43, 42, 43	5	231.57	1.8	6.9	NSGA-II Default
58, 43.43, 46.99, 52, 25.20	5	225.62	3.2	7.91	NSGA-II Default

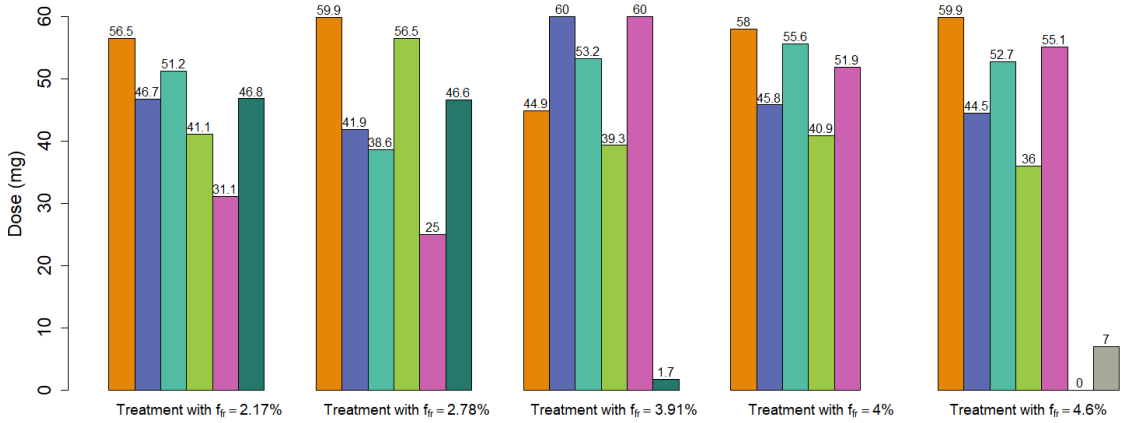

Fig. 3. Barplot representation of some of the treatments with the lowest failure rate after re-evaluation. Each of the bars corresponds to a single day in the treatment.

bars and the failure rate of the treatment is provided below the barplot for that treatment.

What we can observe from Figure 3 is that three out of the five treatments alternate between a very high dosage (between $50mg$ and $60mg$) and a lower dosage. The pattern observed in the top solutions in the previous studies [9], [16] was towards tapered doses where the first dose will be the highest and every dose after it will be lower than the previous. This difference in dose patterns could be explained by the introduction of the PK/PD model as the concentration of antibiotics in the body is modelled differently impacting the $DeathRate_{bacteria}$ calculation.

V. CONCLUSION

The proposed approach looks into the problem of overuse of antibiotics and more specifically optimising the number of antibiotics prescribed and the length of the overall treatment. The automatic design of possible treatments and their evaluation has little constraints at the moment — upper and lower limits on the treatment (3 to 10 days) and upper and lower limits on the daily dosages ($0mg$ to $60mg$) where the bacteria levels are always the same.

In our study, we introduced new techniques to the mathematical model — the PK/PD modelling, and switched from using the standard Gillespie Algorithm to another variation of the Gillespie Algorithm — Tau-leaping for predicting events. Then two population-based multi-objective algorithms were chosen — NSGA-II and SMPSO for designing the antibiotic treatments. Both of the algorithms were then tuned and the hypervolume for each of the runs of the algorithms was calculated. Those hypervolume indicators were then compared as well as some of the best solutions. What could be concluded from the results is that the NSGA-II algorithm provided better results than SMPSO. There was also not a significant improvement upon tuning the algorithms in terms of results at the end of the budget of 5000 iterations. The noise shown in the results proved that it is imperative that the correct algorithm and tuned parameters is used in the optimisation process to guarantee the best solutions.

In this model, the patient's profile (overall health, diet, other possible medical conditions) and correct usage of the antibiotics are not taken into account even though they play a big factor when fighting bacterial infections. These points will be investigated for the future versions of this model as well as including more objectives when designing the treatments.

REFERENCES

[1] Antonio Benitez-Hidalgo, Antonio J Nebro, Jose Garcia-Nieto, Izaskun Oregi, and Javier Del Ser. jmetalpy: a python framework for multi-objective optimization with metaheuristics. *arXiv preprint arXiv:1903.02915*, 2019.

[2] Leonardo CT Bezerra, Manuel López-Ibánez, and Thomas Stützle. Automatic component-wise design of multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3):403–417, 2015.

[3] Joseph M. Cicchese, Elsje Pienaar, Denise E. Kirschner, and Jennifer J. Linderman. Applying optimization algorithms to tuberculosis antibiotic treatment regimens. *Cellular and Molecular Bioengineering*, 10(6):523–535, Dec 2017.

[4] William Jay Conover and William Jay Conover. Practical nonparametric statistics. 1980.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[6] World Health Organization et al. Global action plan to control the spread and impact of antimicrobial resistance. 2017.

[7] Carlos M Fonseca, Luís Paquete, and Manuel López-Ibánez. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE international conference on evolutionary computation*, pages 1157–1163. IEEE, 2006.

[8] Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[9] Iona K. Paterson, Andrew Hoyle, Gabriela Ochoa, Craig Baker-Austin, and Nick Taylor. Optimising antibiotic usage to treat bacterial infections. *Scientific Reports*, 6:37853, 11 2016.

[10] Matthew E Levison and Julie H Levison. Pharmacokinetics and pharmacodynamics of antibacterial agents. *Infectious Disease Clinics*, 23(4):791–815, 2009.

[11] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[12] Cathrine McKenzie. Antibiotic dosing in critical illness. *Journal of antimicrobial chemotherapy*, 66(suppl_2):ii25–ii31, 2011.

[13] Bernd Meibohm and H Derendorf. Basic concepts of pharmacokinetic/pharmacodynamic (pk/pd) modelling. *International journal of clinical pharmacology and therapeutics*, 35(10):401–413, 1997.

[14] Antonio J Nebro, Juan J Durillo, José García-Nieto, Cristóbal Barba-González, Javier Del Ser, Carlos A Coello Coello, Antonio Benítez-Hidalgo, and José F Aldana-Montes. Extending the speed-constrained multi-objective pso (smpso) with reference point based preference articulation. In *International Conference on Parallel Problem Solving from Nature*, pages 298–310. Springer, 2018.

[15] Antonio J Nebro, Juan José Durillo, Jose Garcia-Nieto, CA Coello Coello, Francisco Luna, and Enrique Alba. Smpso: A new pso-based metaheuristic for multi-objective optimization. In *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pages 66–73. IEEE, 2009.

[16] Gabriela Ochoa, Lee A Christie, Alexander E Brownlee, and Andrew Hoyle. Multi-objective evolutionary design of antibiotic treatments. *Artificial Intelligence in Medicine*, 102:101759, 2020.

[17] Gabriela Ochoa, Minaya Villasana, and Edmund K Burke. An evolutionary approach to cancer chemotherapy scheduling. *Genetic Programming and Evolvable Machines*, 8(4):301–318, 2007.

[18] Christopher M Parry, Vo Anh Ho, Phan Van Be Bay, Mai Ngoc Lanh, Le Thanh Tung, Nguyen Thi Hong Tham, John Wain, Tran Tinh Hien, Jeremy J Farrar, et al. Randomized controlled comparison of ofloxacin, azithromycin, and an ofloxacin-azithromycin combination for treatment of multidrug-resistant and nalidixic acid-resistant typhoid fever. *Antimicrobial agents and chemotherapy*, 51(3):819–825, 2007.

[19] Andrei Petrovski, Siddhartha Shakya, and John McCall. Optimising cancer chemotherapy using an estimation of distribution algorithm and genetic algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 413–418, 2006.

[20] Andrei Petrovski, Bhavani Sudha, and John McCall. Optimising cancer chemotherapy using particle swarm optimisation and genetic algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 633–641. Springer, 2004.

[21] Sui-Man Tse, Yong Liang, Kwong-Sak Leung, Kin-Hong Lee, and Tony Shu-Kam Mok. A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):84–91, 2007.

[22] Minaya Villasana and Gabriela Ochoa. Heuristic design of cancer chemotherapies. *IEEE transactions on evolutionary computation*, 8(6):513–521, 2004.

[23] Minaya Villasana, Gabriela Ochoa, and Soraya Aguilar. Modeling and optimization of combined cytostatic and cytotoxic cancer chemotherapy. *Artificial intelligence in medicine*, 50(3):163–173, 2010.