Training Plan Evolution based on Training Models

Conference Paper · September 2015

DOI: 10.1109/INISTA.2015.7276739

Alexander Asteroth
Bonn-Rhein-Sieg University of Applied Sciences
80 PUBLICATIONS **770** CITATIONS
SEE PROFILE

Melanie Ludwig
Bundesinstitut für Arzneimittel und Medizinprodukte
15 PUBLICATIONS **122** CITATIONS
SEE PROFILE

Training Plan Evolution based on Training Models

David Schaefer, Alexander Asteroth, Melanie Ludwig
Bonn-Rhein-Sieg
University of Applied Sciences
53757 Sankt Augustin, Germany
{david.schaefer1,alexander.asteroth,melanie.ludwig}@inf.h-brs.de

Abstract—**Training models have been proposed to model the effect of physical strain on fitness. In this work we explore their use not only for analysis but also to generate training plans to achieve a given fitness goal. These plans have to include side constraints such as, e.g., maximal training loads. Therefore plan generation can be treated as a constraint satisfaction problem and thus can be solved by classical CSP solvers. We show that evolutionary algorithms such as differential evolution or CMA-ES produce comparable results while allowing for more flexibility and requiring less computational resources. Due to this flexibility, it is possible to include well known principles of training science during plan generation, resulting in reasonable training plans.**

I. Introduction

Training models aim at modeling the adaptation of the human body to physical exercise. In general, models take the training load of an athlete as well as parameters characterizing the response of the respective individual into account. Since the 70s, numerous models have been proposed for endurance training (e.g., [1]–[4]). Correlations between real and modeled progress have been observed in a number of studies, varying in significance but altogether pointing out sufficient potential for real world use. Currently models are used mainly as a tool by professional trainers as well as by amateur athletes.

We propose using automatically generated optimized training plans based on training models with individually identified parameters. With such a system an athlete could be assisted to reach his or her goals even without access to a professional trainer, as long as the goal is feasible. An even bigger advantage of these individualized training plans may be the capability of regenerating the training plan after an imposed off time, e.g., caused by illness. The idea of model based plan generation emerged briefly after training models were proposed but, since classical methods were used to solve the resulting optimization and planning problem, usage of results was limited to very constrained situations such as tempering.

We suggest using more flexible approximative approaches like genetic algorithms to solve this problem. This will enable the inclusion of important features such as the definition of constraints on the training plan to be generated. Constraints can be time based, such as certain weekdays which shouldn't be available for exercise, or they could constrain the generated plan to adhere to certain training philosophies.

II. Related Work

The first and still most widely used mathematical model of performance was proposed in 1976 by Calvert, Banister, et al. [1]. In the following years modifications have been proposed, e.g., by Behncke [2] or Busso et al. [3]. All of these models try to model the training-performance relation on an abstract level by two antagonistic variables called *fitness* and *fatigue*. While models based on more physiological details have also been proposed [5], they have not proven useful in practice. During the last years further modifications of the Banister models have been proposed by Allan and Coggan [6], Skiba [7] and Perl [4], [8] with a focus on practical usability and the inclusion of phenomena like *overtraining*.

Nowadays, mathematical models of performance are used by professional trainers to plan the athletes training. In 2013 Clark et al. [9] described that models are used in this context as a diagnostic tool as well as a didactic tool to help trainers learn how to build a reasonable training plan.

In 1991 Fitz-Clarke, Morton and Banister [10] used a model theoretic approach to "invert" the model to find an optimal *tapering*-strategy. More recently Busso and Thomas [11] have criticized the approach due to the strong simplification made therein.

Perl proposes the use of genetic algorithms [8], [12] to find individual parameters of his PerPot model and to optimize a training plan. Unfortunately, no details on results could be found.

III. Training

Due to the importance for this work, in this section we will describe basic priciples used by the models as well as their mathematical details.

A. Training effects

There are three principles in training that are adressed by the mathematical training models used in this study:

1) *Supercompensation*
2) *Overtraining*
3) *Plateau Effect*

The models *Fitness Fatigue* and *PerPot* will be explained later on in this section.

The reason why training can increase the performance of an athlete is the way the human body reacts to a training impulse. First, the body reacts with fatigue or even exhaustion. But during the time of recovery it not only reaches its prior performance level but goes beyond that and keeps that elevated level for a certain amount of time [13]. This principle is called *Supercompensation* and it is one of the most basic principles

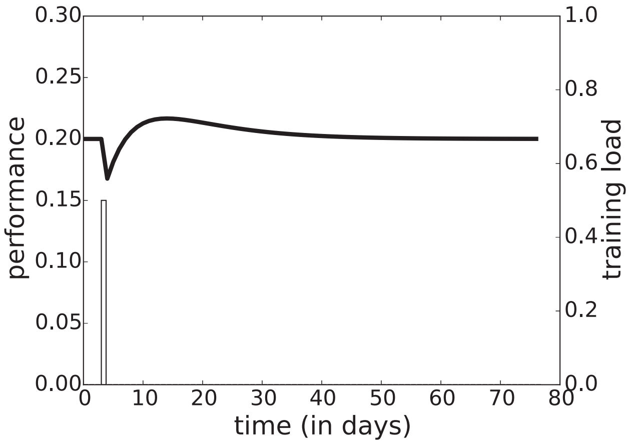

Fig. 1. Supercompensation modeled by Fitness Fatigue. After a single training impulse the body first reacts with fatigue, then with a temporary increase of performance.

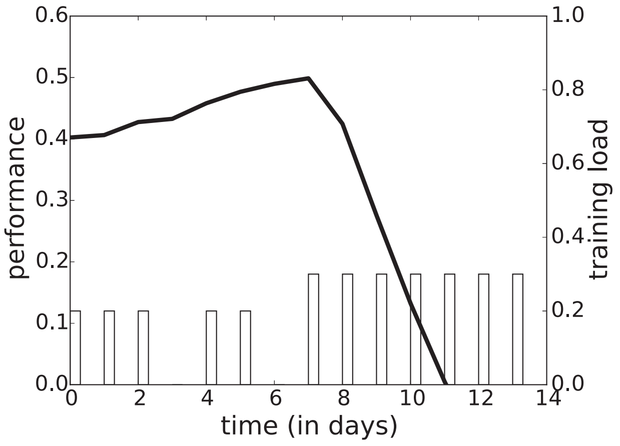

Fig. 2. Overtraining modeled by PerPot. As training load gets too high the body reacts with a sudden and huge decrease of performance.

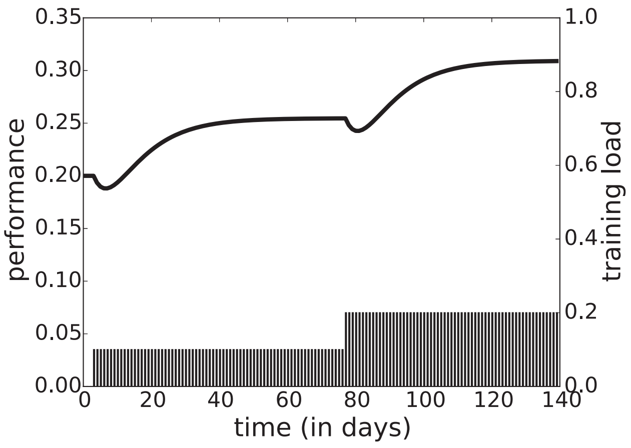

Fig. 3. Plateau Effect modeled by Fitness Fatigue. An ongoing increase of training load is needed to keep increasing performance. Otherwise a plateau is reached and performance stagnates.

of training science. If no new training impulse follows, the performance level decreases again to or below the initial level as can be seen in Figure 1. In order to keep the elevated level of performance or to increase the level again, a continuous effort of training is indispensable.

Caution is needed if an athlete wants to keep increasing his or her performance level. Naturally, there is an individual limit of performance a human can reach. More important in our case: there's also a limit of how much training an athlete can endure. Beyond this limit an effect called *overtraining* occurs. If a human body does not receive enough time to recover from training, it reacts with a negative adaptation in performance and the opposite of the desired effect takes place, as can be seen in Figure 2. Thus overtraining should be avoided at all cost [14].

Another aspect to consider is that performance will reach a plateau if training load is kept constant. In order to keep increasing performance an ongoing increase in training load is needed. Of course, that assumes that the maximum of performance has not yet been reached. But this is rarely the case with non-professional athletes. Both models simulate this behaviour very well as can be seen for Fitness Fatigue in Figure 3.

Additionally there are aspects of practicality our system has to consider in order to be useable in practice. Presenting a training plan to an athlete which claims to lead her to her goal but is incompatible to well known training philosophies is unlikely to be convincing enough to be followed. To be useful for a human athlete a training plan needs to do more than just be optimal with respect to a mathematically defined model. Very short training sessions which make sense with regard to the quality of the approximation but are just impractical for the athlete should be avoided. Thus basic principles like gradually increasing the training load over the course of the whole training plan should be supported by the plan generating system.

B. Load and Performance Metrics

To quantify the load of a training session several definitions for load metrics were proposed. The basic idea of these metrics is to rate the intensity and the time of a training session:

$$load = intensity \cdot duration$$

Among cyclists metrics based on heart rate (HR) or power data are common. An example often used in practice is the heart rate based TRIMP [15], defined as:

$$\text{TRIMP} = D \cdot (\Delta\text{HR ratio}) \cdot e^{b(\Delta\text{HR ratio})}$$

with D being the duration of the training session in minutes, $b = 1.67$ for females, $b = 1.92$ for males, and

$$\Delta\text{HR ratio} = \frac{\text{HRex} - \text{HRrest}}{\text{HRmax} - \text{HRrest}}$$

where HRrest is the average heart rate during rest, HRex is the average HR during exercise and HRmax is the maximum heart rate of the athlete.

To complement load metrics there are a number of performance metrics in use, most of them based on power data. An example is FTP (Functional Threshold Power) which can be explained as the maximum power a cyclist is able to maintain continuously for 60 minutes.

In general, training models are defined using arbitrary units so that they are useable with various metrics for performance and training load. Inputs and outputs of the models are determined by the metrics. The process of tuning the model parameters to an individual requires load and performance data in some units. In this way these units become the units for predicting the performance and prescribing the loads in the model. So if TRIMP and FTP are used to tune the model parameters for an individual, TRIMP and FTP are the units of the model in this case.

C. Training Models

The *Fitness Fatigue* model, aka the Impulse Response model, is widely used in training science [1]. The model tries to predict the adaptation of the human body to training with the simple concept of

$$Performance = Fitness - Fatigue$$

with $Fitness$ representing the positive adaptation effect of training and $Fatigue$ the negative one. The discrete mathematical representation of this concept is the following:

$$\begin{aligned} p(t) &= p^* + k_1 \cdot g(t) - k_2 \cdot h(t) \\ g(n) &= \sum_{t=1}^{n-1} w(t) \cdot e^{\frac{-(n-t)}{\tau_1}} \\ h(n) &= \sum_{t=1}^{n-1} w(t) \cdot e^{\frac{-(n-t)}{\tau_2}} \\ w &: t \rightarrow workload \end{aligned}$$

with g modelling the *Fitness*, h modelling the *Fatigue* and $w(t)$ providing the training workload at time t. The model parameters p^*, k_1, τ_1, k_2, τ_2 need to be fit to the individual athlete. The arbitrary factors k_1 and k_2 and their unit depends on the used metrics for performance and training load, τ_1 and τ_2 are time constants: τ_1 the time needed for the body to react to a training session with an increased performance, and τ_2 describes the duration of fatigue due to the training session [4].

One of the six parameter sets we use in this work is:

p^*	k_1	τ_1	k_2	τ_2
0.2	0.15	10	0.13	6

A shortcoming of the model is the missing concept of overtraining. In fact, inside the model, performance can be increased indefinitely [16]. The next presented model will correct this flaw.

The *Performance Potential Metamodel* (PerPot) claims to be a meta model, a model of models. It is an antagonistic model that simulates a positive and a negative effect of training analogical to Fitness Fatigue. In order to model the delay of the adaptation process PerPot stores the effects in potentials and uses changing flow rates to simulate the increase or decrease of performance. The version that was used in this work has three potentials, the strain potential (SP), storing the negative effect of training; the response potential (RP), storing the positive effect of training and the performance potential (PP), representing the performance level of the modeled athlete.

The calculation of PP at time t_n is based on the state at time t_{n-1} and can be structured into three steps. First, the rise of the potentials SP and RP according to LR (Load Rate) at time t_n:

$$\begin{aligned} &LR : \text{t} \rightarrow \text{workload} \\ &SP := SP + LR \\ &RP := RP + LR \end{aligned}$$

Second, the calculation of flow rates between the effect potentials SP and RP to PP:

$$\begin{aligned} SR &:= \frac{min(min(1, SP), max(0, PP))}{DS} \\ RR &:= \frac{min(min(1, RP), min(1, 1 - PP))}{DR} \\ OR &:= \frac{max(0, SP - 1)}{DSO} \end{aligned}$$

Third, the update of the potentials SP, RP and PP

$$\begin{aligned} SP &:= SP - SR - OR \\ RP &:= RP - RR \\ PP &:= PP + RR - SR - OR \end{aligned}$$

The flow rates are slowed by athlete specific delay values:

$$\begin{aligned} DS &:= \text{Delay of Strain Rate } (SR) \\ DR &:= \text{Delay of Response Rate } (RR) \\ DSO &:= \text{Delay of Strain Overflow Rate } (OR) \end{aligned}$$

Again, these values and the initial performance potential PP need to be fit individually. The potentials and flows are scaled to the interval $[0, 1]$. The scaling factors that are applied to the original load and performance data during the fitting process need to be adapted to the individual as well. Thus, the scaling factors are also part of the individual parameter set [12]. PerPot models overtraining with a second negative effect on performance. If the strain potential reaches a level above 1, the overflow rate rises above 0 and is subtracted from the performance potential [4]. An example for a PerPot parameter set is:

DS	DR	DSO	PP
6.8	6.3	1.5	0.2

IV. Methods

Finding the optimal training plan leading to a specific performance goal is an optimization problem. In this context the search space Ω of the problem consists of all possible training plans of the desired length, with typical training plans lasting over several months. Thus Ω is defined as

$$\Omega = \{l^n | l \in \mathbb{R}_+, n = \text{plan length in days}\}.$$

A. Constraints

We want to support 4 basic constraints with our system:

1) micro cycles
2) off weeks
3) off days
4) max loads

A micro cycle is the smallest building block that is used to define a training plan. In this work we define a micro cycle as the structure of one week in the plan, categorizing the weekdays in days with or without training load. An example is (Monday, Tuesday, Wednesday, Friday, Saturday), meaning that there are loads > 0 on these days. Off weeks and off days are just weeks and days with training loads equal to 0. A max load constraint limits the maximum possible load per day of a plan.

B. CSP-Solvers

Our problem of finding an optimal training plan that adheres to some constraints can be interpreted as a Constraint Satisfaction Problem (CSP). A CSP consists of three components X, D, C, with $X = \{X_1, \ldots, X_n\}$ being the n-element set of variables and $D = \{D_1, \ldots, D_n\}$ being the equally big set of domains. C is the set of constraints, specifying allowable combinations of values.
A single domain D_i contains the allowable values for variable X_i. A single constraint C_i is a suitable constraint representation for the solutions of the problem, for example a tuple of an unallowed value combinations. Defining a problem in such a generic way makes it possible to work on the problem with generic algorithms instead of designing a specific algorithm for every specific problem [17].

In our case, the daily training loads can be defined as X and the possible load values correspond to D. Of course our supported constraints like off days or max load correspond to C. Interpreting our problem in this way yields the possibility to use a number of CSP-solvers and to compare their solutions to other techniques.

C. Evolutionary Approaches

Our second approach will be to use evolutionary algorithms. The fitness f of a solution should mainly depend on the quality of the approximation. We can formalize this as the difference between the given goal and the result a model predicts for the solution. The second aspect we want to consider is the total training load of a solution, which is the sum of all training loads. A plan that approximates a goal equally well as another plan but does so with less training should be preferred. To fight the danger of rating a solution with lower training loads and bad precision higher than a solution with higher training loads and good precision we should weight the total training load with an appropriate factor. Therefore we define f as follows:

$$f(p, g) = 1 - |g - f_m(p, p_m)| - \alpha \cdot \sum_{i=1}^{n} (p_i) \qquad (1)$$

with $p \in \Omega$, g being the desired goal, f_m the function implementing the model and p_m being the model parameters. In our experiments setting the weighting factor $\alpha = 0.001$ showed good results.

As a first naive approach to the problem of finding an optimal training plan we use a *Hill Climbing* approach which operates on a population size of only one. For our case the initial solution x_0 is chosen to be a plan of the desired length n that only contains training loads of 0.0. The mutation operator is implemented by three functions mut_0, mut_1 and mut_2. mut_0 increases the training load on a single randomly chosen day. mut_1 chooses randomly a weekday d and increases the training load on d in every week of the plan. mut_2 simply increases the training load on every day throughout the plan. To support constraints like off days or a maximum daily training load the functions can be configured to adhere to such user defined constraints.

To tackle the problem of local optima our next evolutionary approach makes use of *Differential Evolution* (DE) which is especially suited for real-valued problem domains and not as prone to fall into local optima as Hill Climbing. The strength of DE comes from its so-called DE-operator which works on the differences of individuals to create a new solution out of 4 individuals in the current generation. A feature of DE is the automatic scaling of its step size stemming from the gradual growing concentration of the population in a subspace of Ω, decreasing step sizes as an optimum is approached [18].

Evolution Strategies (ES) are particularly well suited for nonlinear problems in $\mathbb{R}^n$ if dimensionality is not too high. One of the most successful implementations of ES is *CMA-ES* (Covariance Matrix Adaptation-Evolution Strategies) which we choose as our fourth approach for plan generation. In CMA-ES new solution candidates are sampled from a multivariate normal distribution which ensures unbiasedness. To direct the search the covariance matrix of the distribution is adjusted according to the distribution of decent directions [19]–[21] making solutions of higher fitness more likely. Unlike other evolutionary algorithms which often need a lot of parameter tuning, CMA-ES controls the adaptation of the covariance matrix and the step size by itself making it a parameter-free approach and thus very stable, easy to use, and good for comparison. To conduct our experiments we use the *cma* python package [20], [22]. CMA-ES minimizes its objective function while our fitness function (1) was developed to be maximized. By subtracting the function value from one we derive a function to be minimized from (1):

$$\begin{aligned} f_{\text{cma}}(p, g) &= 1 - f(p, g) \\ &= |g - f_m(p, p_m)| + \alpha \cdot \sum_{i=1}^{n} (p_i) \end{aligned}$$

D. *Experiments*

We conduct a number of basic experiments to evaluate the appropriateness of the proposed optimization methods. In addition we carry out experiments to see how the models behave with regard to various basic training philosophies. We observe the behaviour of the models when we change the plans to adhere to some basic training principles. The changes can be made either after plan generation or during the plan generation itself. In the first case we are interested in changes of the predicted performance. With a reasonable model we expect to see positive changes with a refined plan compared to the unrefined version. In the second case (during plan generation) we are interested in the runtime changes of the algorithms as well as a possible decline in approximation quality.

We use six sets for the Fitness Fatigue (FF) model and seven sets for the PerPot (PP) model that were published in [4]. The chosen parameter sets showed a correlation coefficient of $>$ 0.6 in [4] regarding the modeled performance progress and the real measured one of the participating athletes. Unfortunately the PerPot parameters lacked information about the used DSO values, so we weren't able to model overtraining in our experiments with PerPot and set OR to 0 unconditionally. In the following experiments plans with a length of 12 weeks are used, aiming at an increase of performance from 0.1 to 0.3. Each experiment is repeated 10 times.

After the basic experiments in our first modification the sequences of training loads in already generated well approximating plans are sorted in ascending order to see if these postprocessed plans lead to a higher performance compared to the unsorted plans. A training sequence is a number of consecutive days with a non-zero training load. A week in the plan defined as p = [0.2, 0.1, 0.3, 0.0, 0.5, 0.1, 0.0] would be transformed to $p^{'}$ = [0.1, 0.2, 0.3, 0.0, 0.1, 0.5, 0.0].

The second experiment is a counterpart of the first experiment: we sort the sequences in descending order and have a look at the modeled performance change. The idea behind descending training loads is the desire to start a sequence of training days with very intense intervals for which an athlete should be reasonably fresh and well rested.

In our third experiment we verify if DE is still able to work well if we constrain the possible solutions to plans which have sorted sequences. To constrain the solutions in such a way we simply sort every generated solution candidate in the demanded manner.

Our fourth experiment focuses on the plateau effect. In order to avoid the plateu effect it is necessary to construct a plan with increasing loads. We sort the loads of the generated plans in such a way, that the off days keep their position in the plans. So an DE-generated unsorted plan p = [0.2, 0.1, 0.3, 0.0, 0.5, 0.1, 0.0, 0.2, 0.1, 0.6, 0.0, 0.8, 0.3, 0.0] is transformed into $p^{'}$ = [0.1, 0.1, 0.1, 0.0, 0.2, 0.2, 0.0, 0.3, 0.3, 0.5, 0.0, 0.6, 0.8, 0.0]. We expect the models to predict an increased performance for the sorted plans as they are capable of simulating the plateauing effect.

To see if it is possible to support the sorting of the whole plan as a constraint with DE, as our fifth experiment we again simply sort every generated solution candidate in the required manner.

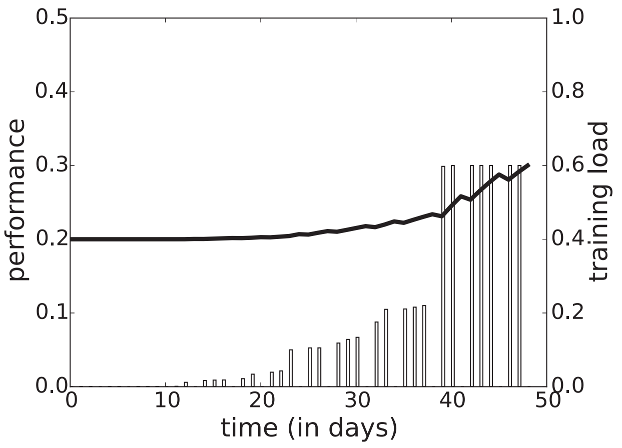

Fig. 4. A plan generated by DE with sorting as the only refinement.

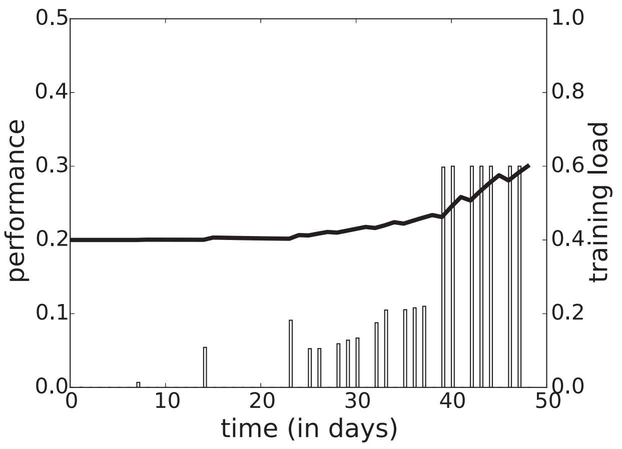

Fig. 5. A plan generated by DE with accumulation of unpractical small loads.

In our last experiment we consider the practicality of a generated plan. Very small training sessions can make sense with regard to the approximation of the given goal in the model but can feel very impractical to the athlete. As a possible solution we propose a postprocessing mechanism for generated plans which tries to maintain a minimum load of in a training session to $x\%$ of the maximum load of the plan. All loads in a week which are below this limit are summed up, and this sum added to the lowest load of a week which is still above the limit. Then those loads which are too small are replaced with off days. An example is given in Figure 4 and Figure 5. There are a number of impractically small training sessions in the first few weeks which can be combined into only two training sessions, and still reach the goal with high precision. In practice, of course, the athlete should set x according to his or her preference, for our experiment we set $x = 10\%$.

V. Results

A. Experimental Results for the Optimization Methods

All experiments we conducted to test our methods were repeated ten times for each of the parameter sets. This means 60 experiments for Fitness Fatigue and 70 experiments for PerPot. For both models the goal was to go from an initial performance level of 0.1 to 0.3 in a 12-week plan. We used the following constraints:

- off weeks = [3, 7, 11]
- micro cycle = (Monday, Tuesday, Wednesday, Friday, Saturday)
- max load = 1.0

We define the quality of approximation of a generated plan p as follows:

$$quality = \frac{f_m(p, p_m)}{g} \cdot 100\%$$

If a plan p leads to a performance level $>$ g, the quality will be $> 100\%$. If p leads to a performance level $<$ g, the quality will be $< 100\%$. Defining the quality of an approximation in this way makes it possible to see if a plan p overreaches the given goal g.

Out of the many CSP-solvers we evaluated, only *Bonmin* (Basic Open-source Nonlinear Mixed INteger programming) and *Couenne* (Convex Over and Under ENvelopes for Nonlinear Estimation) are able to work with the Fitness Fatigue model in a reasonable way. In fact, they are able to find precise solutions for all Fitness Fatigue parameter sets while supporting all kinds of constraints. Both Bonmin and Couenne are based on the Branch-and-Bound-Paradigm which makes it possible to handle search spaces which are otherwise too big for an exhaustive search [23], [24]. None of the CSP-solvers was able to work with the PerPot model, as its definition makes use of the min and max operators, which even Bonmin and Couenne cannot handle.

Using the previously described mutation operators mut_0 and mut_2 Hill Climbing is able to find a solution for four out of the six Fitness Fatigue parameter sets with a deviation of $<$ 5% while respecting all constraints. In the other two parameter sets a local optimum is reached leading to an approximation of only 57.57% and 77.48%. With mut_1 a deviation of $<$ 5% from the goal is measured for five of the parameter sets. Optimization problems with local optima are a known weakness of the Hill Climbing algorithm [25]. So these results are unsurprising.

For the PerPot model it is possible to reach a good approximation ($<$ 5% deviation) for all parameter sets but one, while using mut_0 and satisfying all kinds of constraints. Switching to mut_1 or mut_2 doesn't help for the one failing case. With these mutation operators the algorithm ends up in local optima, too.

In all 60 experiments for the Fitness Fatigue model DE is able to find a solution of good approximation while satisfying all constraints, such as max load, off weeks and micro cycles. The same positive result is achieved for the PerPot model in all 70 experiments, again while satisfying all types of constraints.

Like DE, CMA-ES is able to find near perfect solutions for all Fitness Fatigue and all PerPot parameter sets while supporting all varieties of constraints.

B. Experimental Results for the Support of Training Principles

In our first experiment Fitness Fatigue showed only a small preference for sequences sorted ascendingly as the numbers in Table I show. In comparison, PerPot shows a strong preference for sorted sequences.

For sequences sorted in decreasing magnitude (experiment 2) we can observe a small performance decline in Fitness Fatigue. This behaviour is consistent with the first experiment. The behaviour of PerPot is consistent as well. We can observe a significant decrease of performance.

When sorting sequences during plan generation (experiment 3) for Fitness Fatigue the ability of DE doesn't suffer. There are some cases of decreased approximation quality, but over all the needed number of generations is fewer with sequence-sorted solution candidates, see Table II. With PerPot the number of needed generations increases significantly compared to the same experiment without the constraint of sorted training sequences. However there is no negative effect on the quality of approximation. We can conclude that it is possible to introduce sequence sorting as a constraint into the solution finding process itself.

	FF avg. change $\pm \sigma$	FF max. change	PP avg. change $\pm \sigma$	PP max. change
sequence sorting asc.	0.35% $\pm$ 0.51%	2.3%	5.6% $\pm$ 0.08%	35%
sequence sorting dsc.	-0.36% $\pm$ 0.55%	-3%	-10.6% $\pm$ 10.12%	-43%
sort whole plan asc.	8.8% $\pm$ 11.15%	59.9%	68.6% $\pm$ 17.99%	90.8%
accumulation of small loads	-3.58e-6% $\pm$ 2.58%	-1.2%	0.91% $\pm$ 2.58%	13.6%

TABLE I
CHANGES IN THE MODELED PERFORMANCE AFTER ADAPTING THE GENERATED PLANS TO BASIC TRAINING PRINCIPLES. THE REFINEMENT IS DONE AFTER PLAN GENERATION.

	FF change in needed generations	PP change in needed generations
seq. sort asc. during generation	0.95	1.7
seq. sort dsc. during generation	0.95	3.26
sort whole plan during generation	0.9117	7.52

TABLE II
ADAPTING THE SOLUTION CANDIDATES TO TRAINING PRINCIPLES DURING THE OPTIMIZATION PROCESS OF DE LEADS TO SOME SIGNIFICANT CHANGES IN THE NUMBER OF NEEDED GENERATIONS.

Sorting the whole plan after plan generation (experiment 4) both models show the expected performance increase compared to the unsorted plan. The changes in PerPot are again stronger than in Fitness Fatigue.

Working only with sorted solution candidates, as in experiment 5, speeds up the convergence process of DE using the Fitness Fatigue model. On the other side, the number of needed generations increase significantly for PerPot. In neither model does the approximation quality suffer.

The negative effect we can observe in experiment 6 for Fitness Fatigue is not significant enough to justify ommiting this feature. In PerPot there is even a performance gain on average.

C. Evaluation

To evaluate our results we carry out an exhaustive search by discretization of the search space. We limit the dimensions in the space of possible plans to values between 0.0 and 1.0 with a step size of 0.1. We can support this in our system with a max load constraint. Further, we choose one parameter set per model and set the length of the plan to 7 days. This way we end up with a space of $11^7 = 19487171$ possible plans for both models. Of course this will only give us an approximation of the real optimum but since there are no feasible analytical methods for deriving the optimum it at least gives us a baseline validation for our results by comparing the elements to a generated plan p.

There are basically two things we can evaluate about our plan generation system:

1) Is a generated plan p optimal or is there a plan $p^{'}$ which approximates the given goal g better?
2) Does a generated plan p lead to the given goal g with the least possible overall training load or is there a plan $p^{'}$ that leads to g as well but with less load?

For both models we search a plan that leads to the performance goal $g = 0.2$ in 7 days, starting at performance level 0.1.

Exhaustive search for the Fitness Fatigue model yields 88 plans approximating g with a quality of $100\% \pm 4.966 \times 10^{-5}\%$. We see, even in this relatively small search space a large set of possible solutions exists.

For comparison we generate 10 solutions with DE. These plans approximate g with a quality of $99.945\% \pm 0.2661\%$. It follows, even this very restricted exhaustive search can find plans which approximate g with higher accuracy than DE. The downside is a runtime 3000 times the length of DE. CMA-ES and the CSP-solvers Bonmin and Couenne are able to find a precise solution in a fraction of the time the exhaustive search needs. So CMA-ES and the two CSP-solvers are the best techniques regarding the approximation of a solution. DE is preferable to the exhaustive search as it finds solutions with a precision that can be judged as sufficient in practice and also needs just a fraction of the runtime.

Concerning the overall load of a solution the CSP solvers, again, are better than DE while CMA-ES is only second to Couenne. That the solution of Couenne is even better than the best solution of the exhaustive search shouldn't be taken into account as a real exhaustive search would be able to find the Couenne solution as well.

	exhaustive search	DE	CMA-ES	Bonmin	Couenne
approximation quality mean $\pm$ σ	100% $\pm$ 4.966e-05%	99.945% $\pm$ 0.2661%	100% $\pm$ 0.00%	100% $\pm$ 0.00%	100% $\pm$ 0.00%
min. overall load	3.0	3.0567	2.9181	3.0072	2.9177
avg. runtime	3015.53s	1s	37.6	16,1s	16,8s

TABLE III
EVALUATION RESULTS FOR FITNESS FATIGUE

	exhaustive search	DE	CMA-ES
approximation quality mean $\pm$ σ	99.9928% $\pm$0.1246%	100.0009% $\pm$0.0208%	100.0000% $\pm 8.5641e-06\%$
min. overall load	0.4	0.4287	0.4048
avg. runtime	4961.05s	1s	17.6s

TABLE IV
EVALUATION RESULTS FOR PERPOT

Exhaustive search for PerPot yields 11 plans in the discretized search space approximating g with an average quality of $99.9928\% \pm 0.1246\%$. Again we see, even in this very limited search space a big set of possible solutions exists.

The comparison with DE is based on 10 generated solutions with the same goal and constraints. DE is able to approximate g with an average quality of $100.0009\% \pm 0.0208\%$ So DE can surpass the precision of the restricted exhaustive search and looking at the runtime numbers DE is the more practical solution as well.

The 10 solutions generated by CMA-ES show a near perferct precision and have a better minimal overall load than DE, see Table IV. Like DE, CMA-ES also needs just a fraction of the time of the exhaustive search.

VI. Conclusion

In the previous sections we presented possible approaches to automate the generation of training plans based on mathematical training models. Two antagonistic models, Fitness Fatigue and PerPot, were used to estimate the effect on an athletes' fitness induced by a plan. Phrased as an optimization problem with side constraints, the problem was successfully solved by classical CSP solvers as well as evolutionary algorithms DE and CMA-ES. Due to the use of functions not provided by CSP solvers, only one of the models was successfully used to generate plans. The average runtime of CSP solvers was significantly higher than that of evolutionary approaches.

Evolutionary algorithms found results of comparable quality as CSP solvers Bonmine and Couenne in significantly shorter time, with Differential Evolution (DE) being faster than CMA-ES but resulting in lower approximation quality. Early stopping CMA-ES might yield comparable performance as DE.

Both algorithms were able to generate near optimal plans for all models considering all side constraint that were defined.

For all approaches it was possible to include constraints such as predefined off days and off week as well as basic training principles. Including these principles into plan generation resulted less computation time in case of the Fitness Fatigue model. The plans generated using ascending training intensities gained higher increases in performance with less overall intensity.

All models used in this study are antagonistic models that define performance as the difference between fitness and fatigue. This reduction in information was criticized in recent publications which lead to new, two valued models. Maximizing fitness while simultaneously minimizing fatigue is a multi objective optimization problem. In future research we will use evolutionary algorithms such as NSGA-II or PAES to solve this MOOP.

The results we achieved so far are purely based on computer simulations and therefore not validated in practice. A next step will be to evaluate generated training plans in real training and record progress in performance. The acquired data will serve as a basis to adjust the parameters in the underlying models, and thus the generated plans, to better reflect the observed performance gain.

References

[1] T. W. Calvert, E. W. Banister, M. V. Savage, and T. Bach, "A Systems Model of the Effects of Training on Physical Performance," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 2, pp. 94–102, Feb. 1976. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5409179

[2] H. Behncke, "A mathematical model for the force and energetics in competitive running," *Journal of mathematical biology*, pp. 853–878, 1993. [Online]. Available: http://link.springer.com/article/10.1007/BF00168050

[3] T. Busso, R. Candau, and J.-R. Lacour, "Fatigue and fitness modelled from the effects of training on performance," *European journal of applied physiology and occupational physiology*, vol. 69, no. 1, pp. 50–54, 1994.

[4] M. Pfeiffer, "Modeling the relationship between training and performance-a comparison of two antagonistic concepts," *International journal of computer science in sport*, vol. 7, no. 2, pp. 13–32, 2008.

[5] A. Mader, "Aktive belastungsadaptation und regulation der proteinsynthese auf zellulärer ebene," *Deutsche Zeitschrift für Sportmedizin*, vol. 41, no. 2, pp. 40–58, 1990.

[6] H. Allen and A. Coggan, *Training and racing with a power meter*. VeloPress, 2010.

[7] P. Skiba, "Evaluation of a novel training metric in trained cyclists," *med. sci. sports*, vol. 39, 2007.

[8] J. Perl, "Antagonistic adaptation systems: An example of how to improve understanding and simulating complex system behaviour by use of meta-models and on line-simulation," in *16th IMACS Congress*, 2000.

[9] D. C. Clarke and P. F. Skiba, "Rationale and resources for teaching the mathematical modeling of athletic training and performance." *Advances in physiology education*, vol. 37, no. 2, pp. 134–52, Jun. 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23728131

[10] J. R. Fitz-Clarke, R. Morton, and E. Banister, "Optimizing athletic performance by influence curves," *Journal of Applied Physiology*, vol. 71, no. 3, pp. 1151–1158, 1991.

[11] T. Busso and L. Thomas, "Using mathematical modeling in training planning," *International journal of sports physiology and performance*, vol. 1, no. 4, p. 400, 2006.

[12] J. Perl, "Perpot: A metamodel for simulation of load performance interaction," *European Journal of Sport Science*, vol. 1, no. 2, pp. 1–13, 2001.

[13] J. Ivy, "Muscle glycogen synthesis before and after exercise," *Sports Medicine*, vol. 11, no. 1, pp. 6–19, 1991. [Online]. Available: http://dx.doi.org/10.2165/00007256-199111010-00002

[14] S. Israel, "Zur Problematik des übertrainings aus internistischer und leistungsphysiologischer Sicht." *Med. u. Sport*, vol. 16, pp. 1–12, 1976.

[15] E. Banister, "Modeling elite athletic performance," *Physiological testing of elite athletes*, pp. 403–424, 1991.

[16] J.-P. Brückner, "Training im Leistungssport: Modellierung und Simulation von Adaptationsprozessen," Ph.D. dissertation, Kiel, Christian-Albrechts-Universität, Diss., 2007, 2009.

[17] S. Russell and P. Norvig, *Artificial Intelligence - A modern approach*. Pearson, 2010.

[18] R. Storn and K. Price, *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995, vol. 3.

[19] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE, 1996, pp. 312–317.

[20] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.

[21] A. Auger and N. Hansen, "Tutorial cma-es: evolution strategies and covariance matrix adaptation." in *GECCO (Companion)*, 2012, pp. 827–848.

[22] N. Hansen, "Cma evolution strategy source code," May 2015. [Online]. Available: https://www.lri.fr/~hansen/cmaes_inmatlab.html#python

[23] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya *et al.*, "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optimization*, vol. 5, no. 2, pp. 186–204, 2008.

[24] E. M. Smith and C. C. Pantelides, "A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex minlps," *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 457–478, 1999.

[25] J.-M. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE, 1994, pp. 312–317.