

EcoRacer: Game-based Optimal Electric Vehicle Design and Driver Control Using Human Players

Yi Ren*

Mechanical Engineering
Arizona State University
Tempe, Arizona, 85287
Email: yiren@asu.edu

Alparslan Emrah Bayrak

Mechanical Engineering
University of Michigan
Ann Arbor, Michigan, 48109
Email: bayrak@umich.edu

Panos Y. Papalambros

Mechanical Engineering
University of Michigan
Ann Arbor, Michigan, 48109
Email: pyp@umich.ed

ABSTRACT

We compare the performance of human players against that of the Efficient Global Optimization (EGO) algorithm for an NP-complete powertrain design and control problem. Specifically, we cast this optimization problem as an online competition and received 2391 game plays by 124 anonymous players during the first month from launch. We found that, while only a small portion of human players can outperform the algorithm in the long term, players tend to formulate good heuristics early on that can be used to constrain the solution space. Such constraining of the search enhances algorithm efficiency, even for different game settings. These findings indicate that human-assisted computational searches are promising in solving comprehensible yet computationally hard optimal design and control problems, when human players can outperform the algorithm in a short term.

1 Introduction

Optimal design problems can still challenge our computational ability to solve them. A case in point is the design of an electric vehicle combining topology (configuration) design, proportional design and control design for overall system optimization [1]. This problem is NP complete, not amenable to an all-in-one solution and with a disjoint feasible domain.

*Address all correspondence to this author.

Other computationally hard design problems include material synthesis, drug design, and mechanical, electrical and structural topology design [2–7].

More recently, “human computation” [8] has been reported as a promising alternative to solving tough optimization problems. A seminal experiment was the Foldit [9] game, which gathers large numbers of non-expert players (a “crowd”) to perform protein structure prediction by minimizing the protein’s energy with spatial rearrangement of its structure. Since its launch in 2009, Foldit has attracted 300,000 players and has shown an advantage of human spatial reasoning for pursuing near-optimal solutions [10]. A follow-up study showed that the most popular folding strategy derived from the crowd is comparable in performance to an expert-developed computer algorithm for protein folding [11]. Inspired by this success, Lee et al. tackled the challenge of RNA synthesis, a combinatorial optimization problem [12], with the eteRNA experiment [13] showing that a human crowd with no knowledge of the underlying science improved their problem solving skills, outperformed existing computer algorithms, and even contributed knowledge to the creation of a more effective algorithm. Le Bras et al. proposed a way to expedite search in combinatorial problems by identifying the optimal values for a subset of variables through human-computer interactions [14]. They demonstrated that a material discovery problem can be cast as an online game solved visually by human players. The Phylo game is another example that utilized human pattern recognition capability in solving an NP-hard optimization problem called Multiple Sequence Alignment, where players are set to identify similar genome sequences of animals by moving colored blocks around [15].

These results indicate the potential of using human intuition (sampling good design solutions [10, 12] and recognizing visual patterns [14, 15]) and human intelligence (learning and generating design rules [11, 12]), for solving challenging optimization problems or enhancing numerical optimization solvers. Gamification, i.e., using games purposefully [16], is one way to engage large numbers of human participants in such tasks. The above studies do not provide analysis on whether investment in gamification is more cost-effective than improving existing computer algorithms. However, it appears that such “citizen science” can be valuable when (a) a group of trained human solvers can be gathered and maintained, and (b) efficient solution of a *large number* of problems of similar mathematical nature is desired. At this point, the implementations of these existing games are case-dependent, and whether their individual success can be replicated in solving other problems remains an open question [17].

These successes and doubts regarding human computation motivated this paper. How much can gamification offer in solving computationally hard design optimization problems?

This paper investigates the potential benefits of using human crowd computation through games designed to address the aforementioned electric vehicle design problem. The basic idea is to “learn” (in computer science terminology) the problem’s constrained solution space from successful and failed player attempts, and to search adaptively for a near-optimal solution within this constrained space. This idea is an extension of apprenticeship learning (see [18] for an example), where the machine apprentice follows or explores similar actions as the expert master.

The rest of the paper is structured as follows: We introduce the motivating problem of vehicle powertrain design and control in Section 2 and its gamification in Section 3. In Section 4, we look into computational solutions to the game problem with and without using information from human players. Section 6 compares performance of human players against the

computational solution, and discusses the use of human computation as well as some technical details of the experiment. We conclude with Section 7.

2 Motivating Problem

A growing number of civil [19–22] and military [23–25] vehicle applications are being considered for electrification due to ever restrictive emission reduction and energy security requirements. With different driving conditions (speed and power demands), vehicle optimal powertrain design and control strategies can be quite different [26]. Therefore, efficient design automation is desired to identify optimal solutions (i.e., the powertrain design and the control strategy) given the input vehicle specifications and driving conditions.

Fathy et al. [27] and Peters et al. [28] investigated the combined design and control problem, and showed that ignoring the coupling between design and control yields suboptimal solutions. One approach to handle the coupling is to formulate an all-in-one optimization problem to include both design and control [29, 30] problems in a single one; another approach is to nest optimal control within the design problem. This latter approach is more commonly used in hybrid electric vehicle configuration studies [1, 31–33], where existing optimal control algorithms such as Dynamic Programming (DP) [21, 34], Pontryagin’s Minimum Principle (PMP) [35, 36] and Equivalent Consumption Minimization Strategy (ECMS) [37] can be applied. The DP method finds the control strategy with globally optimal energy efficiency by discretizing the state space and using optimal control to find a shortest path; PMP and ECMS have lower computation cost but can find only local optimal solutions. They require some model simplifications to guarantee global optimality [36]. The computational cost of DP grows exponentially with the size of the state space and the number of time steps [38]. The search for the optimal powertrain design and control policy can therefore become expensive, especially when a large variety of topology design candidates exist.

Optimal control algorithms commonly used for vehicle control require full information of the mechanical and electrical behavior (models), as well as future driving conditions, i.e., probability distributions of speed and power demand. When this information is (partially) unknown, learning mechanisms are required to guide the control strategy. In particular, when repeated trials are allowed, such as during the design phase, apprenticeship (imitation) learning algorithms [18, 39] are developed for the controller to follow sample trajectories demonstrated by human experts. This paper is different from existing reinforcement learning studies in two aspects: (1) While existing work either utilizes an existing human expert [39] or learns a strategy (perhaps implicitly through a reward function) natural to human beings [18], here we examine a problem that is hard for individual players and is solved through crowd competition; (2) rather than developing a learning algorithm, here we focus on understanding the efficacy of human computation.

3 The ecoRacer Game

We introduce the ecoRacer game [40] that translates the powertrain design and control problem into a crowd competition. The game asks players to drive an electric car through a given track. Unlike traditional racer games that compete on speed, players must spend as little energy as possible to complete the track in 36 seconds¹. To achieve this goal, players must employ

¹This time limit is calibrated empirically so that the game is challenging enough to encourage replays, but not too difficult to drive new players away.

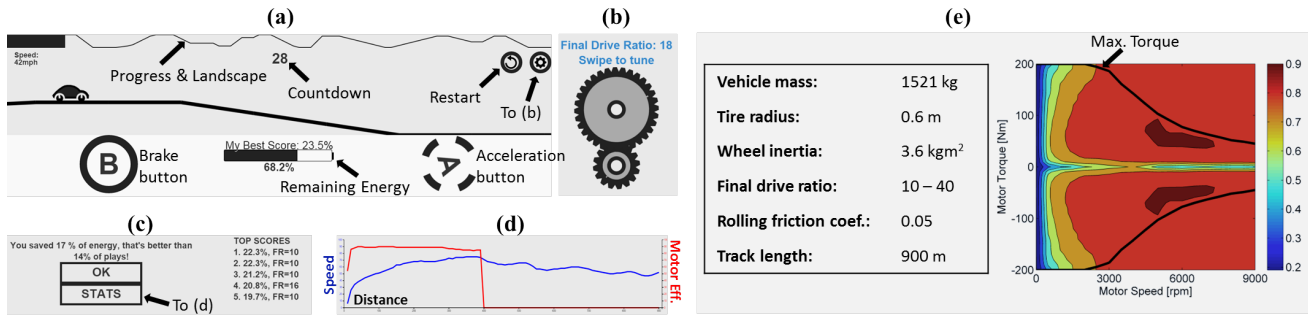


Fig. 1. (Color online) The ecoRacer game interface (a-d) and settings (e). The game interface (a) is designed to work for computer and mobile screens. Touching the gear icon to the right enters the design screen (b), where the final drive ratio can be tuned. After each play, the score board (c) is shown, and the player can review speed and motor efficiency profiles of the current play in (d). Vehicle specifications, motor efficiency map, and maximum torque curves are shown in (e).

good control strategies in combination with selection of a final drive ratio - a design variable. The final drive ratio range is set between 10 and 40. A higher ratio will result in a larger output torque to the front wheels but lower maximum vehicle speed. After each completion of the track, the battery energy consumption is submitted as the player's score, and players are notified about their current ranking among all competitors. The leading scores along with their final drive ratios are also shown to the players. These features follow existing trendy games and we hypothesize that they will trigger competition among players, leading to more plays of the game. However, the hypothesis is not tested in this paper. Game features are summarized in Figure 1.

The control involves only acceleration and braking to ensure that the game is intuitive and easy to play on mobile platforms and for all audiences, including those with no driving experience. The controls are presented to the player at the beginning of the game. Energy consumption (regeneration) by acceleration (braking) is visualized on a battery bar. The Chipmunk 2D physics engine [41] is incorporated to model the track, car body, motor, wheels and suspensions, with supplied motor efficiency map, maximum and minimum torque curve, and other vehicle parameters. The physics engine simulates the scene once every 1/48 second, allowing precise control of the car by the player. For mobile devices with lower computing capability, this high frequency simulation may lead to slower car movement. Since the clock counts in real-time, the players on these devices become less competent. In order to address this issue, we defined "one second" in the game as the time spent on completing 48 simulation calls.

4 Computational Solutions of the ecoRacer Game

In order to benchmark the performance of human players, this section provides two computational solutions to the ecoRacer game. The first considers the game as a nested optimization problem: The outer loop searches for an optimal final drive ratio, while the inner loop solves a dynamic programming problem to find the optimal control policy for the given design. Through a nearly exhaustive search of the discrete control and design space we can obtain a solution close enough to the true global optimum. We should emphasize that this approach requires all models and parameters that constitute the game to be known. In contrast, the second approach treats the game as a black-box function that inputs control and design variables and outputs a score. The black-box function is learned based on existing trials through metamodeling (response

surfaces). This latter approach is closer to how we usually solve optimization problems where simulations or experiments are involved; it is also a fair comparison to human players as both the machine and humans learn and improve through playing the game.

4.1 Optimal design and planning with full knowledge

Let P_{batt} be the instant battery power consumption, $\omega_{mot}(t)$ and $T_{mot}(t)$ the motor speed and torque at time t , v the vehicle speed and a the vehicle acceleration. Also denote by t_f the time spent on finishing the race, $x(t_f)$ the distance covered at t_f , x_{final} the total track length, v_{max} the maximum speed limit, and a_{max} and a_{min} the limits on the vehicle acceleration. The objective is to minimize the battery consumption denoted by E_{batt} with respect to the final drive ratio ρ and the control of $T_{mot}(t)$, while completing the race within the time limit t_{max} . Note that t_f is not a free variable but it depends on the decision variables ρ and $T_{mot}(t)$. This minimization problem can be formulated as follows:

$$\begin{aligned}
& \min_{\rho, T_{mot}(t)} E_{batt} = \int_0^{t_f} P_{batt}(\omega_{mot}(t), T_{mot}(t), \rho, t) \cdot dt \\
& \text{subject to } t_f \leq t_{max} \\
& x(t_f) = x_{final} \\
& v(0) = 0 \\
& 0 \leq v(t) \leq v_{max} \\
& a_{min} \leq a(t) \leq a_{max}.
\end{aligned} \tag{1}$$

The vehicle speed and acceleration are determined by motor speed ω_{mot} and motor torque T_{mot} following:

$$\begin{aligned}
v(t) &= \omega_{mot}(t) R_{tire} / \rho \\
a(t) &= \frac{T_{mot}(t) \rho / R_{tire} - F_{res}}{M_{veh} + 2 J_{wheel} / R_{tire}^2},
\end{aligned} \tag{2}$$

where R_{tire} is the wheel radius, J_{wheel} the wheel inertia, M_{veh} the vehicle mass, and F_{res} the resistive force, including both road resistance and extra load from road inclination.

We solve the energy minimization problem in Equation (1) using a nested formulation described below.

4.1.1 Inner loop control problem

The inner loop control problem for a given design is solved by DP. Specifically, we discretize the track into $N = 180$ equal steps, each corresponding to 5 meters. This setting provides an accurate enough solution while keeping a single call of the DP algorithm computationally affordable. We use the horizontal position and speed of the vehicle as state variables, and the change of speed, Δv , as the decision variable. At each step i , Δv can take three values that correspond to the three actions players can take during the game: $\Delta v_{max,i}$ for acceleration, $\Delta v_{min,i}$ for brake, and 0 for no action. The values of $\Delta v_{max,i}$ and $\Delta v_{min,i}$ are determined by the motor torque limits based on the current state. To incorporate the time constraint, we first

calculate time cost along with energy consumption for each decision at every state, and then treat the violation in the total time span as a penalty in the DP objective. With these assumptions, the DP formulation in Equation (1) for given ρ becomes:

$$\begin{aligned}
& \min_{\Delta v_i} \sum_{i=1}^N E_{batt}(x_i, v_i, \Delta v_i) + \lambda \sum_{i=1}^N t(x_i, v_i, \Delta v_i) \\
& \text{subject to } x_1 = 0 \\
& \quad x_N = x_{finish} \\
& \quad v_1 = 0 \\
& \quad 0 \leq v_i \leq v_{max} \quad \forall i = \{1, 2, \dots, N\} \\
& \quad \Delta v_i \in \{\Delta v_{min,i}, 0, \Delta v_{max,i}\} \quad \forall i = \{1, 2, \dots, N\} \\
& \quad \sum_{i=1}^N t(x_i, v_i, \Delta v_i) \leq t_{max},
\end{aligned} \tag{3}$$

where $\lambda \geq 0$ is a penalty weight: Larger λ results in smaller t_f . We use the bisection method to find the minimum λ that satisfies the time constraint. Note that this requires solving the DP problem multiple times.

4.1.2 Outer loop design problem

The outer loop design problem searches for an optimal final drive ratio that minimizes battery energy consumption. The problem is discretized since players are only allowed to choose from a discrete set of final drive ratios. Starting with three initial ρ values, we fit a quadratic function to approximate the fuel consumption and iteratively minimize this function. We obtain convergence when two subsequent iterations give the same optimal solution. For $\rho > 20$, no λ value yields a control solution to satisfy the time constraint due to low vehicle maximum speed caused by high final drive ratio. The optimal design $\rho^* = 18$ is identified with 47.8% battery state of charge (SOC) upon finishing the track. Depending on the initial set of final drive ratios selected for the outer loop, the inner loop is called four to seven times with each call taking 1.7 to 2 hours to converge to a λ that satisfies the time constraint for the given final drive ratio². Figure 2 summarizes the optimization results. As validation, we tested the same strategy using the game engine and obtained a final battery SOC of 43.4%. The difference between the game engine and the DP calculation is due to (1) the discrepancy between the physics engine and the DP model, e.g., vehicle jumping could happen in the game but is not modeled in DP, and (2) the discretization scheme involved in the DP solution. A finer discretization will reduce the difference with increased computation time.

4.2 Optimal design and control through Efficient Global Optimization (EGO)

We now discuss the EGO algorithm that iteratively learns a good design and control strategy without relying on the settings of the game.

²This simulation is performed using a desktop computer with Intel Xeon E5-2620 CPU clocked at 2.10 GHz and 128GB RAM.

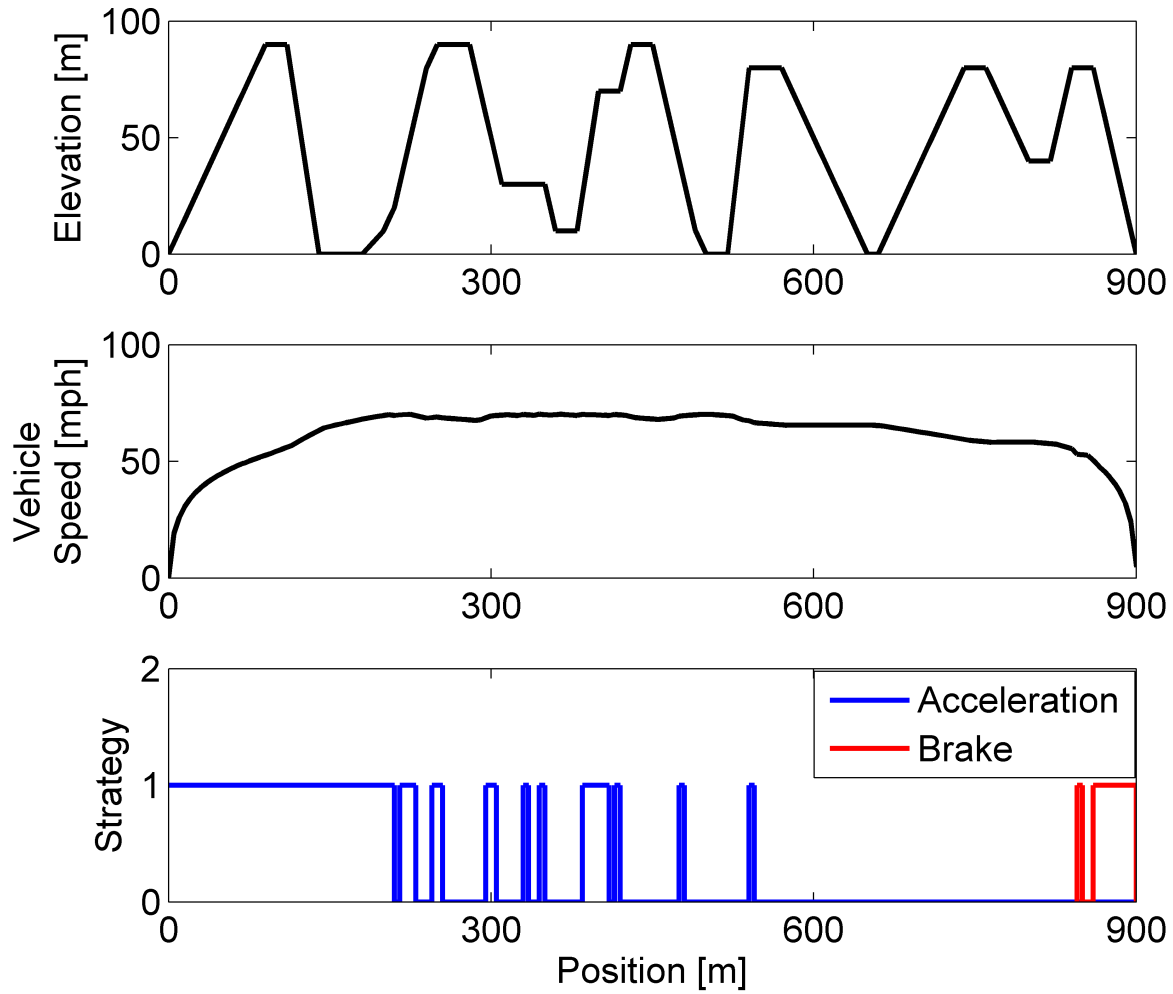


Fig. 2. (Color online) DP results for the optimal final drive ratio. Top to bottom: Road profile, optimal speed profile corresponding to $\rho^* = 18$, and optimal control decisions in terms of braking and acceleration.

4.2.1 Control parameters

For each play by the computer (or human player), we keep track of the control signals for acceleration and braking, denoted by c , along with the following four states: (1) track slope s (“1” for uphill, “-1” for downhill and “0” for flat ground), (2) remaining distance d , (3) remaining time \bar{t} , and (4) vehicle speed v . We assume that the control strategy can be parameterized by some vector \mathbf{w} , so that, given \mathbf{w} , the control signal can be determined by the states:

$$c = \begin{cases} 1 \text{ (Accelerate)} & u(s, d, \bar{t}, v, \mathbf{w}) \geq 1 \\ -1 \text{ (Brake)} & u(s, d, \bar{t}, v, \mathbf{w}) \leq -1, \\ 0 \text{ (No signal)} & \text{otherwise} \end{cases} \quad (4)$$

where $u(\cdot)$ is a mapping from the joint space of states and control parameters to the control signal. Denote the space of \mathbf{w} as \mathcal{W} , which represents a subset of all control strategies that a human player can deploy. The formal determination of $u(\cdot)$ and

\mathcal{W} using human data will be discussed in Section 5. For elaboration on the search algorithm, it suffices to consider \mathcal{W} as a bounded vector space and $u(\cdot)$ as a bounded function defined on \mathcal{W} .

4.2.2 The score

Given a final drive ratio ρ and control parameters \mathbf{w} , the game can be simulated to output the battery charge consumed, denoted by e where $0 \leq e \leq 1$, and the remaining distance from the terminal, denoted by d_{end} . We define the following score as the objective to maximize for optimization:

$$f(\rho, \mathbf{w}) = \mathbb{1}(d_{\text{end}} = 0)(1 - e) - d_{\text{end}}, \quad (5)$$

where $\mathbb{1}(\cdot)$ is an indicator function that returns 1 when its argument is *true*, or 0 otherwise. A successful play where $d_{\text{end}} = 0$ will output the final SOC that is $(1 - e)$, while a failed play outputs the negative remaining distance. This objective favors the completion of the track, but also avoids evaluating failures indifferently.

4.2.3 The EGO algorithm

We can now employ the EGO algorithm, a search routine suitable for optimizing a black-box function defined on a continuous and bounded design space [42]. Denote the solution space as $\mathcal{S} := \mathcal{W} \times \mathcal{D}$, where $\mathcal{D} = [10, 40]$ is the one-dimensional design space. The algorithm starts by sampling \mathcal{S} . It then creates a kriging model using the initial samples and their responses. The next sample is chosen to maximize the expected improvement of the objective. Then, the kriging model is updated by incorporating the new sample and its response. The modeling and sampling procedure is repeated until some termination criterion is met. Figure 3 summarizes the EGO procedure. The nested maximum expected improvement problem is solved using a genetic algorithm.

5 Augmented EGO search using Human Plays

We now discuss parameterization of human plays and how the EGO algorithm can be improved based on these plays.

5.1 Parameterization of human plays

Recall that each play can be represented as a sequence of 5-tuples with control signals and corresponding states. To avoid over-burdening the server, we collect these 5-tuples at each unit distance during the play. We denote the total number of unit distances by N and each recorded play as $\{c_i, s_i, d_i, \bar{t}_i, v_i\}$, for $i = 1, \dots, N$ ³.

Note that \mathbf{w} represents the control strategy to be optimized along with the final drive ratio. For optimization purpose, it is ideal to find a low dimensional \mathcal{W} (and thus \mathcal{S}) where \mathbf{w} s can be found to preserve the recorded control signals according to Equation (4). In practice, we manually tune the definition of $u(\cdot)$ so that replaying the best human play, i.e., with the highest recorded score, using the converted control parameters will yield a score close to the original one. This manual process leads

³The scores of replays using these recorded data are not the same as the original scores from players. This is because, while the game takes player inputs once every 1/48 seconds, we only record a subset of these signals at discrete distance values.

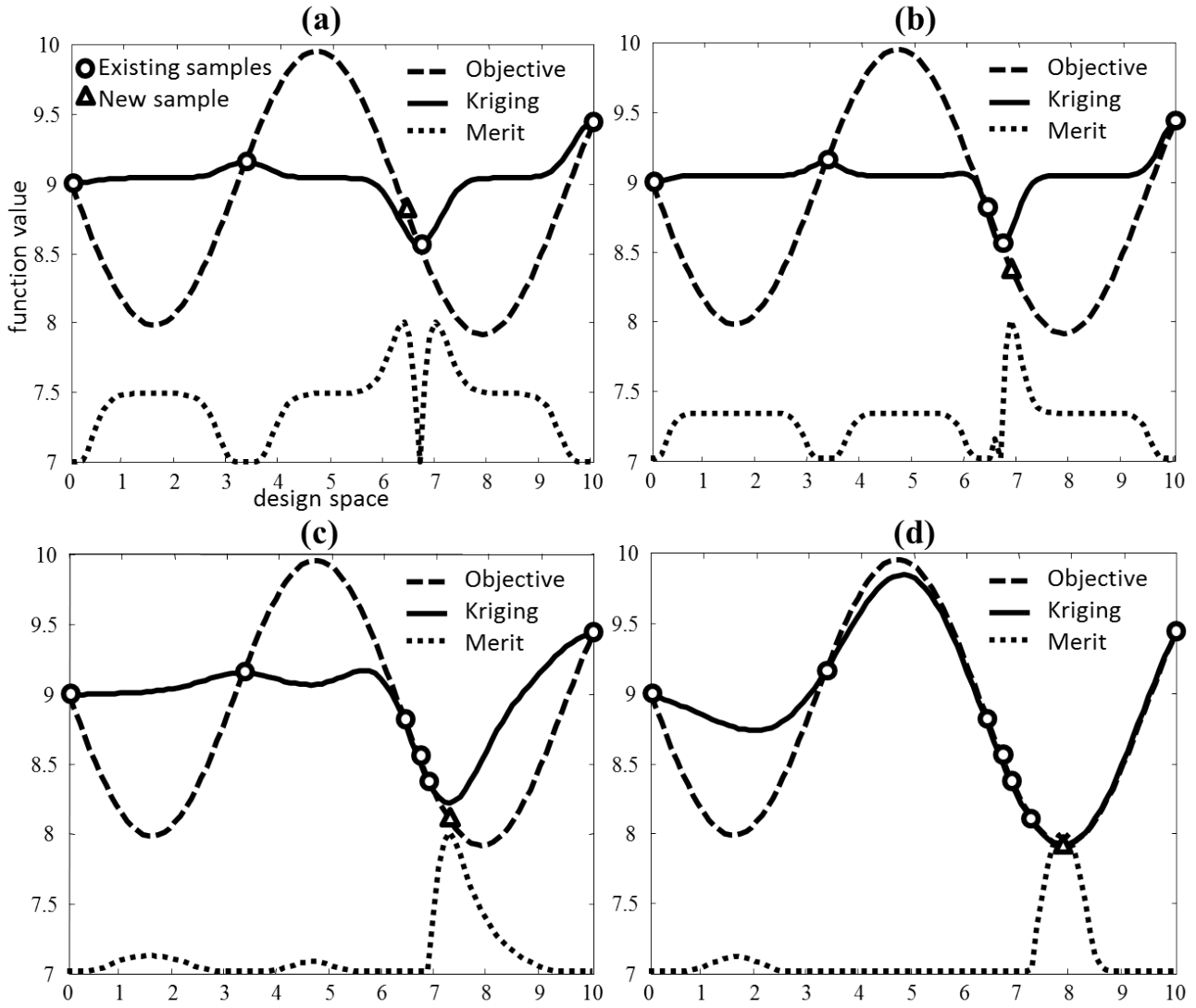


Fig. 3. A one-dimensional example of EGO search (a-d). At each iteration, the algorithm updates the kriging model (in solid line) and chooses the next sample (triangle) according to the maximum expected improvement (merit) function.

to the following choice of $u(\cdot)$:

$$u(s, d, \bar{t}, v, \mathbf{w}) = [s, d, \bar{t}, v, sd, sv, d/\bar{t}, d^2, d^3] \mathbf{w}. \quad (6)$$

For a recorded play, the control parameters \mathbf{w} are chosen by minimizing the discrepancy between the true control signals and

those derived from Equation (4):

$$\begin{aligned}
h(\mathbf{w}) = & \sum_{i=1}^N \max\{(1 - u(s_i, d_i, \bar{t}_i, v_i, \mathbf{w})), 0\}^2 \mathbb{1}(c_i = 1) \\
& + \max\{(1 + u(s_i, d_i, \bar{t}_i, v_i, \mathbf{w})), 0\}^2 \mathbb{1}(c_i = -1) \\
& + \max\{(-1 + u(s_i, d_i, \bar{t}_i, v_i, \mathbf{w})), 0\}^2 \mathbb{1}(c_i = 0) \\
& + \max\{(-1 - u(s_i, d_i, \bar{t}_i, v_i, \mathbf{w})), 0\}^2 \mathbb{1}(c_i = 0).
\end{aligned} \tag{7}$$

We then replay all human plays using the converted control parameters along with their corresponding final drive ratios to update scores.

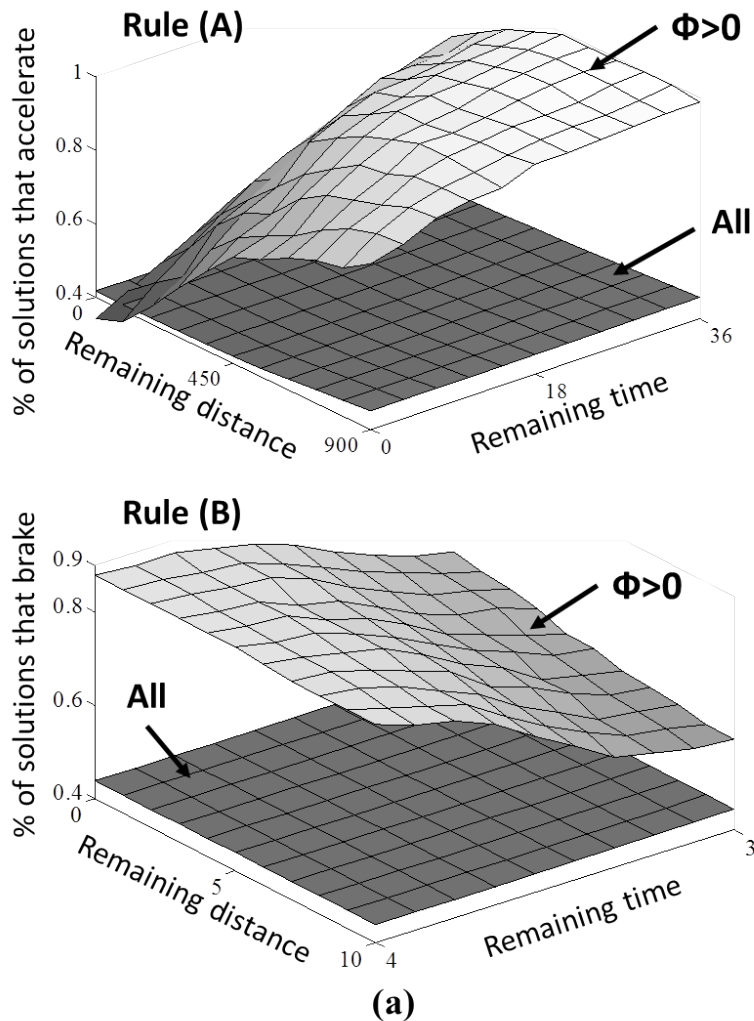
Note that instead of handcrafting the function $u(\cdot)$, we could also use a neural network to learn the mapping between states and control signals, and later consider network parameters as control parameters. However, an arbitrarily defined network could introduce a larger number of control parameters to achieve a good fit to the data, and thereby create a larger search space for the EGO algorithm. For the same reason, a nonlinear kernel (with infinite dimensions) is not used.

In addition, the EGO algorithm requires bounds to be specified for the space of control parameters. Here we identified that the control parameters of the best human play are bounded in $[-3, 3]$. Therefore setting $\mathcal{W} := [-3, 3]^9$ will ensure that the best human solution is available to EGO.

5.2 Classification of human solutions

The parameterization step described above leads to a data set describing all human plays and their corresponding scores. To extract knowledge from these data, we create a one-class classifier for plays with positive scores. The classifier, denoted as $\phi([\rho, \mathbf{w}])$, predicts whether the input solution is likely to succeed ($\phi > 0$) or fail ($\phi \leq 0$), and can be used as a constraint during the search. The classifier is built using LIBSVM [43] with a Gaussian kernel. The kernel parameter γ is set to 0.1 and the training error parameter ν to 10^{-6} . Our rationale for training the classifier with all plays with positive scores rather than the few with top scores is that the former represent a broader range of strategies that *finish* the track. The resultant classifier will thus represent a more relaxed constraint in the solution space than the one derived from only the top plays, offering solution strategies that could be more generally applicable to the ecoRacer game with game settings different from the current game. To empirically demonstrate knowledge learned by the classifier from human plays, we examine two basic control policies that should be applied universally to all game settings: (A) If the vehicle has zero speed when going uphill, it should accelerate in order to complete the track; and (B) if the vehicle is approaching the terminal with a high speed, it should restore energy by braking.

To verify that $\phi > 0$ captures these two, we uniformly draw 10^6 samples from \mathcal{S} . For each sample solution, we calculate its control signals under a variety of states specified in Figure 4. For instance, for rule A, we fix the vehicle speed to zero and the slope to “uphill” while sweeping the other two states across all their feasible values. We consider each solution and states pair as a test point. The percentage of test points that give the correct control signal according to the two rules can be



Rule	(A)	(B)
Slope	Uphill	Downhill
Remaining dist. (m)	0-900	0-10
Remaining time (sec)	0-36	4-36
Speed (m/s)	0	80
$\Phi > 0$	82.5%	76.2%
All	42.9%	44.1%

(b)

Fig. 4. The “learned” subspace of solutions provides more rational control strategies. (a) Percentage of test points that follow Rules (A) and (B) for a variety of states. The percentage is calculated for all uniformly sampled solutions in \mathcal{S} , and for those that satisfy $\phi > 0$. (b) A summary of tested states and percentages averaged across all state values.

calculated among all samples or among the ones that satisfy $\phi > 0$. Results summarized in Figure 4 show that incorporating $\phi > 0$ into the search offers a significantly higher chance of sampling reasonable solutions. The set \mathcal{S} consists of a large number of irrational control strategies, e.g., braking while running out of time, or depleting energy to rush through the track. According to the one million samples, the subspace of $\phi > 0$ accounts for only 0.02% of the entire \mathcal{S} . We note that while certain heuristics (e.g., rule A) can be manually coded into the search by the algorithm designer, others cannot. Taking rule B as an example, while human players would decelerate towards the terminal to regenerate energy, the relationship between the timing of braking and the vehicle states (e.g., speed and distance from the terminal) is difficult to formulate explicitly, especially when neither the environment nor the vehicle model are known.

6 Experiment and Results

We summarize the experiment with human players and compare their performance with that of the EGO algorithm. We then apply the classifier learned from human plays to the same energy optimization problems with different track settings.

6.1 Comparison between human players and the algorithm

The ecoRacer game was announced on Facebook and WeChat on October 14th, 2014, and introduced to a sophomore engineering design class at the University of Michigan on November 4th, 2014. The statistics and analysis in this paper are based on data received before November 26th, 2014. The development team was advised not to play the game during this period. A total number of 124 unique participants registered and played the game, with 2391 plays recorded. The best play (by user “ikalyoncu” on his or her 150th trial) reached a score of 43.2%, only marginally worse than the DP solution of 43.8%. In addition, this player identified the actual optimal final drive ratio of $\rho^* = 18$. Statistics of the game are summarized in Figure 5. Raw player inputs and corresponding scores can be accessed from http://www.public.asu.edu/~yren32/resource/RenPapers/idetc2015game_data.zip.

In the same figure we compare human performance, colored by player IDs, against that of EGO in the first 200 plays. The EGO implementation starts with five initial samples uniformly drawn from \mathcal{S} in the first iteration. Data from five independent EGO runs are collected due to the probabilistic nature of the initial sampling and the genetic algorithm employed to solve the maximum expected improvement problem. For both human and EGO plays, we also plot the averaged best scores. For human players, the calculation at each iteration only considers those who reach that iteration. Lastly, since all failed human plays are stored with scores of zero, we assign failed EGO plays with zeros for fair comparison.

Major findings from the experiment are as follows: (1) One can see that most players quit early and thus are outperformed by the computer in the long run. Only three players have higher scores than the computer after 200 plays. In addition, the fact that only 41% of players (51 out of 124) played more than 10 times suggests that the game is either too hard or not interesting enough for the participants. (2) For dedicated players who kept playing, however, their progress along trials is evident, especially when compared with the EGO results: While the EGO algorithm is designed to balance exploration of the space and exploitation of observations, the 10-dimensional solution space as well as the rugged objective function prevented the algorithm from converging to a local solution in 200 iterations. Although the algorithmic parameters (such as the Gaussian parameter and the merit function form) can be tuned to improve EGO’s performance on a specific problem, dedicated human players provided an alternative way to search with no parameter tuning. Note that the bimodal distribution in scores towards 200 plays, i.e., high scores and zeros, is due to the nature of the game that high scores are achieved when one can finish the track nearly on time by spending just enough energy. Our interpretation is that human players gradually learned this rule, trying to improve their scores while risking failing the game.

These data, along with feedback from some of the players, provide us the qualitative understanding that human players are capable of learning quickly and creating good solutions, but only a few can optimize their solution by precisely executing near-optimal control strategies.

6.2 Search with and without crowdsourced knowledge

The comparison between the EGO algorithm and the players alone showed limited advantage of the latter in finding a good solution in the long run, largely due to the fact that only few players have the necessary persistence to fine tune their strategy by playing repetitively. However, as we show in Figure 6, the method introduced in Section 5.2 allows us to turn

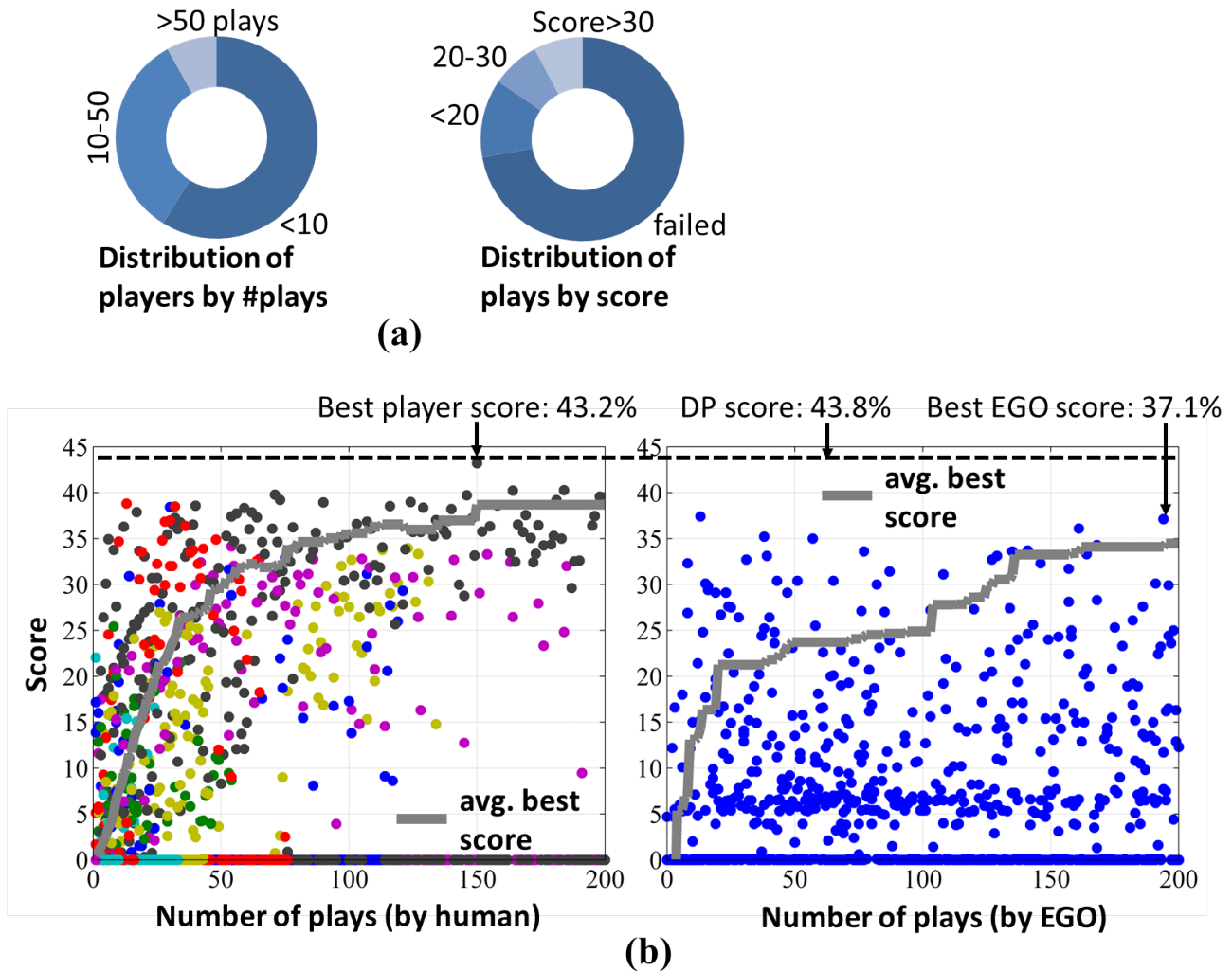


Fig. 5. (Color online) Statistics of the ecoRacer game. (a) Distribution of players by the number of games they played, and distribution of plays by their score. (b) Performance comparison of human player and the EGO algorithm for the first 200 plays.

player data into a constraint in the solution space and to enhance the search for games that were not played by humans.

Specifically, we tested four different tracks to investigate whether the learned knowledge is transferable or scalable: These are the “inverse,” “hill,” “zigzag” and “long” tracks. The “inverse” track is a flipped version of the original track. The “long” track duplicates the original track and the energy capacity five times. For each track, we run the EGO algorithm with and without enforcing the constraint $\phi > 0$ during the search. In the computation, a penalty of $10^6\phi$ is added to the score if the solution is infeasible ($\phi < 0$). In each case, the algorithm starts with five initial samples and terminates after a total of 200 samples. Since ϕ is created using a support vector machine, the initial samples are randomly chosen among the support vectors when the constraint is incorporated, or otherwise randomly sampled from \mathcal{S} .

Figure 6 compares performance from the two algorithm settings, using the average values and standard errors from five independent EGO runs for each case. The result shows that, while EGO without human plays can identify good solutions in the long run, the algorithm can gain a significant early advantage using heuristics extracted from human plays. Further, knowledge learned from human players is transferable to different track settings and scalable to longer tracks, offering a

promising solution to the “curse of dimensionality” [38].

One may naturally then question whether the EGO algorithm could learn from its own plays for future improvements and whether collecting all 2391 human plays is necessary to achieve good heuristics. While this paper does not address these questions theoretically, we used the “long” track to show that for the same small set of human and EGO plays, the former could lead to better future search; with enough plays, the two will have a similar contribution to future search. Figure 7 compares the EGO performance with four different constraints generated from (1) all human plays, (2) all EGO plays, (3) the first 500 human plays, and (4) the first 500 EGO plays, using the original track. The results show a significantly worse performance with the constraint learned from the first 500 EGO plays, while learning from all EGO plays achieves performance comparable to that from all human plays.

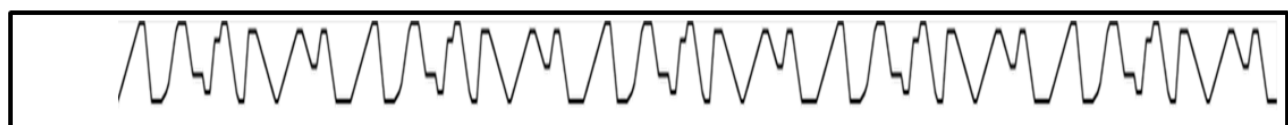
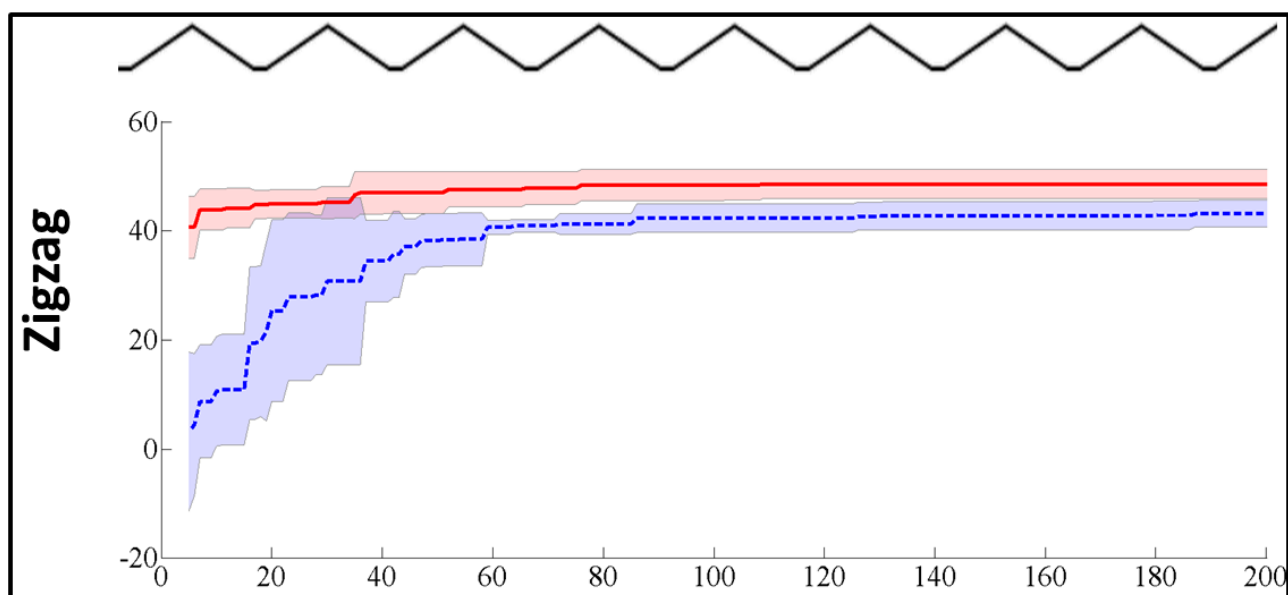
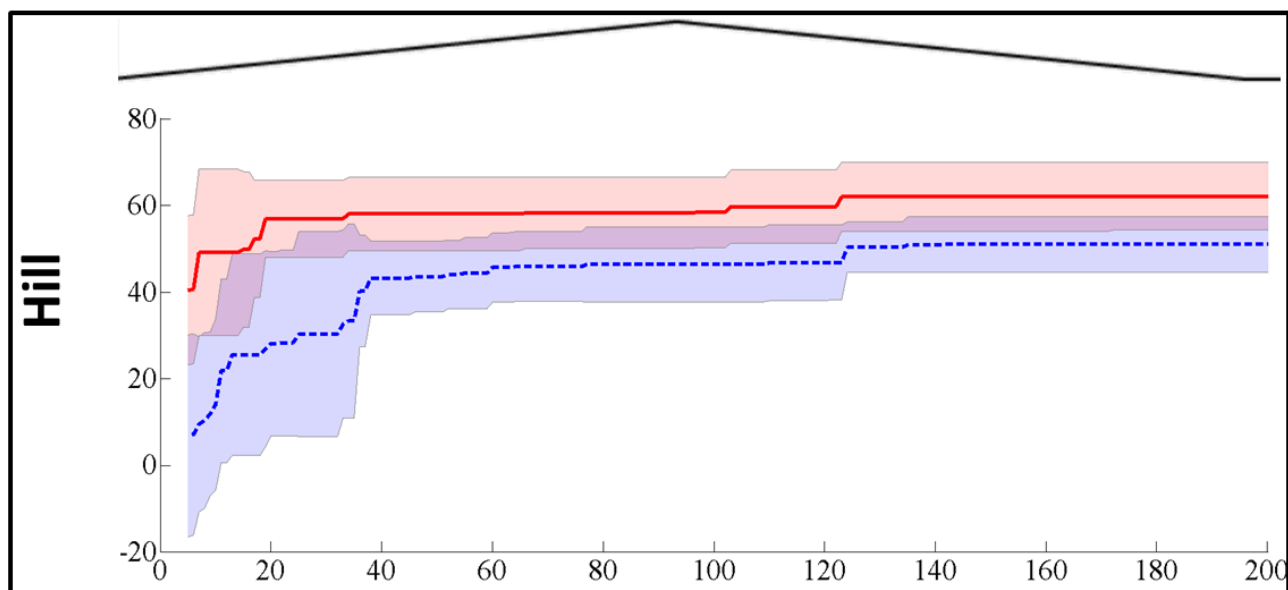
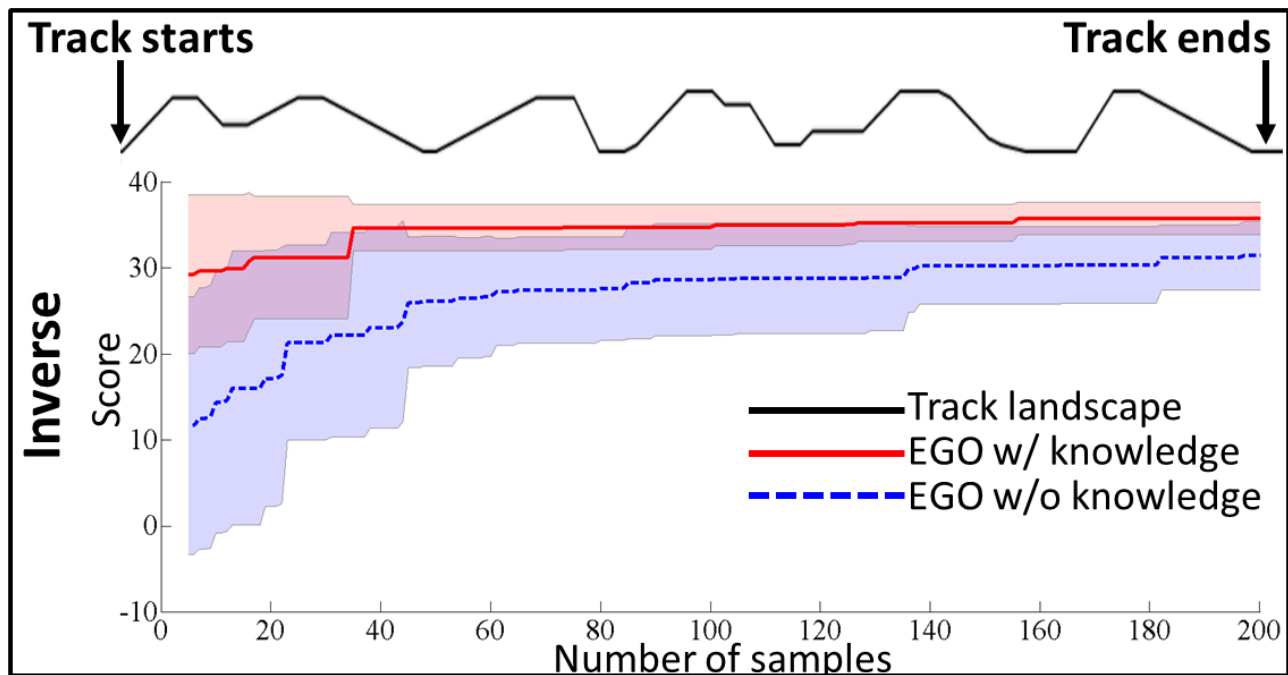
Lastly, learning from too small a set of human plays could diminish search performance. Applying a constraint learned from the first 100 human plays, the averaged score for the “long” track after 200 iterations is 39, significantly lower than that from the previous experiment (Figure 6) with no human plays (60). A closer investigation shows that the constraint classifies all best solutions discovered from unconstrained EGO runs as infeasible. In other words, a small amount of plays with low successful rate could result in a biased conclusion and reject potential solutions, leading to a search algorithm that is worse than one with no prior knowledge. To further verify, we created five classifiers $\hat{\phi}_{100,200,\dots,500}$ from the first 100 to 500 plays and checked whether they can correctly identify the best solutions (obtained from unconstrained EGO runs) for all tracks as feasible. Results are consistent across all tracks: The best solutions discovered from unconstrained EGO runs are identified as feasible only after 400 or more plays are considered.

To understand this finding better, we calculated the success rate of human players, i.e., the rate of finishing the track with non-zero remaining energy, for the first 100 to 500 plays. These rates are 10.0%, 19.0%, 29.3%, 42.3% and 38.0%, respectively. We can then hypothesize that human plays become useful when the players achieve a certain level of success rate, i.e., discover good but not necessarily optimal solutions. However, the rate drops when new players enter with unsuccessful plays. With an increasing number of plays, we observe not only an improvement in the success rate in general, but also in the score. Therefore, determining the number of plays needed for generating a useful EGO constraint based on success rate could be conservative.

6.3 Discussion

The experiment and simulation studies presented here help to answer questions regarding the use of human computation for engineering design.

Why would humans be useful in an optimization task? Human computation, often in the form of crowdsourcing, has mainly been successful in batch tasks that rely on human intuition rather than computational resource, see [8] for examples. Nonetheless, human computation becomes especially beneficial when intuitive tasks are actually computationally expensive. As demonstrated by Foldit and eteRNA, some human players have excellent ability at solving spatial optimization problems, as their search efficiency outperforms that of computer algorithms. The ecoRacer game, similarly, is based on the hypothesis that some players are well trained at tuning their control strategy and design through trial-and-error. The hypothesis is



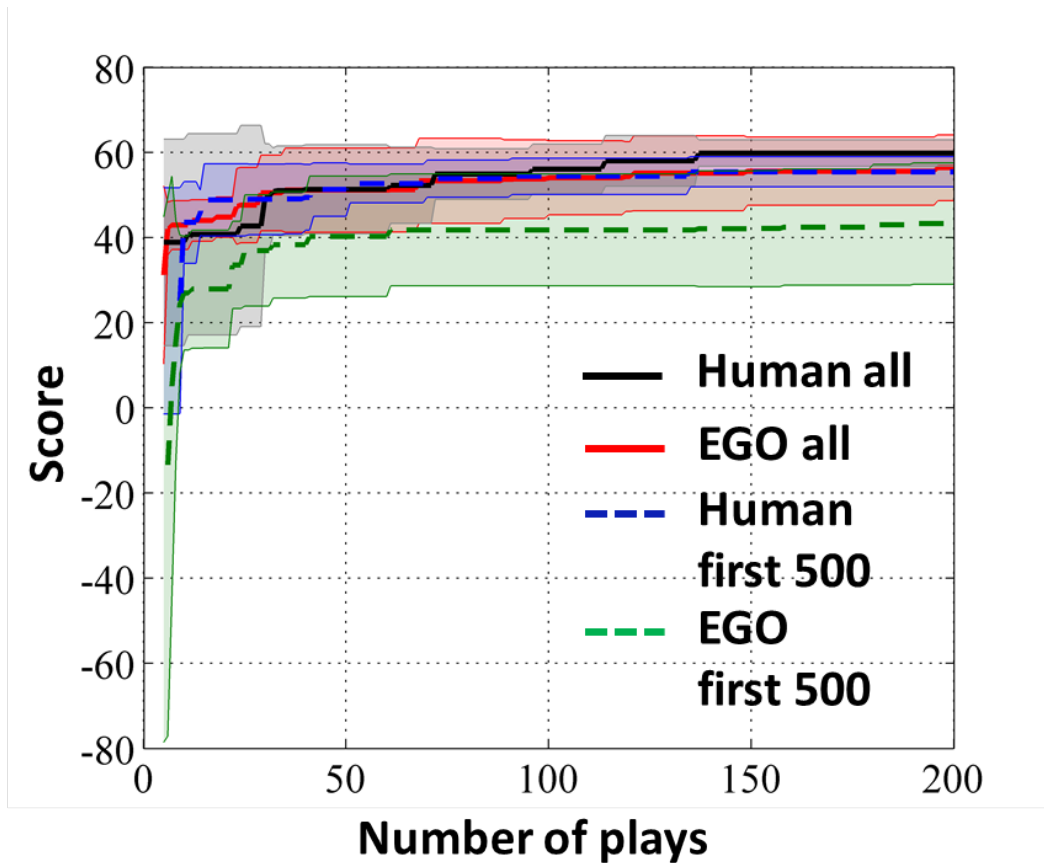


Fig. 7. (Color online) EGO performance on the “long” track with constraints from (1) all human plays, (2) all EGO plays, (3) the first 500 human plays, and (4) the first 500 EGO plays.

reasonable considering that people around the world spend three billion hours training themselves in online games every week [17], and that a large portion of the games requires players to perfect their control. As Figure 5 showed, some players learned good strategies much earlier than the algorithm did. However, the underlying optimization problem must be well translated (and camouflaged) by the game so that people can appreciate the problem and enjoy the fun without facing the actual engineering problem. This requirement leads to the next question.

Is it worth expending the effort of translating any particular problem and even building a game for it, in the hope that some talented people could solve it? Human computation experiments require significant effort to create the proper game environment and to establish sophisticated competition, collaboration and rewarding mechanisms. These efforts are necessary for continuously growing the player population in order to increase the chance of finding talented players. The efforts are not likely to be worthwhile if the underlying problem needs to be solved only once. Given sufficient computational resources, the computer algorithm is likely to outperform human players either by iterating long enough (see Figure 7 in [11] for an example) or by solving the problem via brute-force, e.g., generating the entire tree of states and costs for dynamic programming⁴. Protein folding [10], RNA synthesis [12] and powertrain design for vehicles of particular usage may be tough enough problems requiring repeated solution and thus worthy of human computation investment. In addition, we should emphasize that extracting heuristics from human players and applying them to new problem settings, as demonstrated

⁴While finding a shortest path can be done in polynomial time, the cost of generating the graph could grow exponentially with respect to the track length.

in [11, 12] and this study, could be of significant importance in solving problems facing the curse of dimensionality. As noted in Section 6.2, the machine can also learn from its own experience to effectively constrain its search in the long run. Therefore, human computation has an advantage when human players can outperform the algorithm in the short run. The short-term advantage of human search is that it may capture prior knowledge about the problem that a machine does not acquire. Therefore, future work should investigate the extraction of such human prior knowledge from a limited number of effective solutions from players.

In summary, while we cannot yet compare the cost effectiveness of human computation vs. optimization algorithms, human computation can be valuable for (i) problems comprehensible to people without specialized knowledge, with evidence suggesting that humans find solutions based on their experience that are not discovered by an algorithm; and (ii) problems that must be solved many times with different parameters, so that long-term gains can compensate for the initial game infrastructure investment.

Given that players are more likely to spend their spare time on playing regular games than solving scientific or engineering puzzles, is human computation a realistic alternative? For example, while Foldit attracted 300,000 players, there are hundreds of millions of active users of Angry Birds [17]. Given this reality, even if human computation is proven viable for solving computationally expensive design optimization problems, would there be enough players per problem for the strategy to be effective? Our experiment, as well as existing reports on human computation, showed that the problem can be efficiently solved by a small population of core players. In the case of ecoRacer, it was those players who contributed the successful plays. This finding implies that while the crowd attracted by any particular problem will not necessarily be large, these self-motivated participants may suffice to produce valuable data to expedite problem solving.

A related question is how challenges of translating and camouflaging the problem would be addressed for design problems more complicated than the ecoRacer. We believe that these challenges may not relate to complexity of the design problem, e.g., large problem size, non-convexity, or lack of models. Existing problems, such as in FoldIt, eteRNA, and in this paper, are complicated (NP-complete or NP-hard), yet easily understood by most players. Rather, the challenge is to make the design problem naturally fun to play. Therefore, a significant amount of effort may be needed to calibrate game mechanisms so that the crowd is incentivized, if the problem does not by itself attract players. We have not yet studied whether such mechanisms are effective across problems, regardless of problem complexity.

Some technical issues of the presented ecoRacer experiment are discussed below:

1. From a reinforcement learning perspective, the EGO approach can be categorized as “Direct Policy Search” where the entire simulation is treated as a black-box. All comparisons between EGO and human plays are therefore limited to our current algorithm design. Other popular direct methods such as NEAT [44], as well as indirect algorithms such as actor-critic [45] should be tested and their integration with human play data should be investigated.

2. Converting user control signals to control parameters \mathbf{w} caused discrepancy in the resultant scores. This is because the bases in the mapping $u(\cdot)$ are chosen so that the score derived from *the best* play’s control parameters is close to its actual score. Therefore, the classifier ϕ is not derived directly from the genuine player data. This discrepancy can be reduced by choosing a better set of bases for $u(\cdot)$ that closes the gaps between *each* play’s original score and that from the converted

control parameters.

3. We used a one-class classification that solely extracts knowledge from successful plays. An alternative approach would be to take failed plays into account and derive a binary classifier. It is also possible to create the classifier by utilizing the actual scores, as opposed to the binary “failed” and “successful” labels.

4. The motivation for showing the best scores as well as the drive ratios is to give players an incentive and some hints to further improve their solutions. We believe that such game mechanism is necessary to motivate enough players to participate. However, this mechanism causes the human computation (the game) to be different from the EGO algorithm: (i) The best scores allow players to measure how close are their solutions, based on which they may either fine-tune their strategies or choose to explore new ones; (ii) the displayed drive ratio allows newcomers to quickly converge to a good drive ratio without much exploration, providing them a good initial drive ratio for their search. However, even with information sharing, most players are still out-played by the computer in the long run. In addition, we observe no convergence to the optimal drive ratio value in the first 500 game plays from the crowd. Therefore the conclusions in Section 6.2 are valid even though the mechanisms of the game and the EGO algorithm are slightly different.

7 Conclusion

We examined the value of incorporating game-based human computation in solving optimization problems in the context of an optimal powertrain design and control problem. The results showed that, while only a small portion of human players outperformed the EGO algorithm, useful heuristics can be extracted from the recorded plays and effectively applied to problem settings that were not presented to the players. This indicates the promising use of human computing in transferring knowledge learned from human-comprehensible problems to similar ones of larger scale or difficulty, to achieve scalable and effective search. The findings from this paper offer useful insights for future attempts at solving computationally expensive engineering optimization problems through human computation. In future work, it would be interesting to investigate if useful information can be extracted from the evolution in human plays, and how decomposition strategies involving problem partitioning and coordination that have been effectively applied to numerical solvers can be incorporated within a human computation framework. With a larger crowd of participants, we could also investigate the relationship between the crowd size and its performance. Lastly, it is valuable to explore game mechanisms that encourage players to improve their strategies in more efficient ways.

Acknowledgement

This work has been supported by the National Science Foundation under Grant No. CMMI-1266184. This support is gratefully acknowledged. We thank anonymous reviewers for insightful critique that helped improve this presentation significantly. We also thank members of the Optimal Design Laboratory at the University of Michigan for suggestions and advice on game design and experimentation, and all our players for making this paper possible.

References

- [1] Bayrak, A. E., Ren, Y., and Papalambros, P. Y., 2013. “Design of hybrid-electric vehicle architecture using auto-generation of feasible driving modes”. In Proceedings of the ASME 2013 International Design Engineering Technical Conferences, ASME.
- [2] Larsen, U. D., Signund, O., and Bouwsta, S., 1997. “Design and fabrication of compliant micromechanisms and structures with negative poisson’s ratio”. *Journal of Microelectromechanical Systems*, **6**(2), pp. 99–106.
- [3] Miskin, M. Z., and Jaeger, H. M., 2013. “Adapting granular materials through artificial evolution”. *Nature materials*, **12**(4), pp. 326–331.
- [4] Schneider, G., and Fechner, U., 2005. “Computer-based de novo design of drug-like molecules”. *Nature Reviews Drug Discovery*, **4**(8), pp. 649–663.
- [5] Koza, J. R., Bennett III, F. H., Andre, D., Keane, M. A., and Dunlap, F., 1997. “Automated synthesis of analog electrical circuits by means of genetic programming”. *IEEE Transactions on Evolutionary Computation*, **1**(2), pp. 109–128.
- [6] Bendsøe, M. P., and Kikuchi, N., 1988. “Generating optimal topologies in structural design using a homogenization method”. *Computer methods in applied mechanics and engineering*, **71**(2), pp. 197–224.
- [7] Kicinger, R., Arciszewski, T., and Jong, K. D., 2005. “Evolutionary computation and structural design: A survey of the state-of-the-art”. *Computers & Structures*, **83**(23), pp. 1943–1978.
- [8] Von Ahn, L., 2009. “Human computation”. In 46th ACM/IEEE Design Automation Conference, IEEE, pp. 418–419.
- [9] Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popović, Z., et al., 2010. Foldit. <http://fold.it>.
- [10] Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popović, Z., et al., 2010. “Predicting protein structures with a multiplayer online game”. *Nature*, **466**(7307), pp. 756–760.
- [11] Khatib, F., Cooper, S., Tyka, M. D., Xu, K., Makedon, I., Popović, Z., Baker, D., and Players, F., 2011. “Algorithm discovery by protein folding game players”. *Proceedings of the National Academy of Sciences*, **108**(47), pp. 18949–18953.
- [12] Lee, J., Kladwang, W., Lee, M., Cantu, D., Azizyan, M., Kim, H., Limpaecher, A., Yoon, S., Treuille, A., and Das, R., 2014. “Rna design rules from a massive open laboratory”. *Proceedings of the National Academy of Sciences*, **111**(6), pp. 2122–2127.
- [13] Lee, J., Kladwang, W., Lee, M., Cantu, D., Azizyan, M., Kim, H., Limpaecher, A., Yoon, S., Treuille, A., and Das, R., 2014. eterna. <http://eterna.cmu.edu>.
- [14] Le Bras, R., Bernstein, R., Gomes, C. P., Selman, B., and Van Dover, R. B., 2013. “Crowdsourcing backdoor identification for combinatorial optimization”. In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, AAAI Press, pp. 2840–2847.
- [15] Kawrykow, A., Roumanis, G., Kam, A., Kwak, D., Leung, C., Wu, C., Zarour, E., Sarmenta, L., Blanchette, M., Waldispühl, J., et al., 2012. “Phylo: a citizen science approach for improving multiple sequence alignment”. *PloS one*, **7**(3), p. e31362.

- [16] Von Ahn, L., 2006. "Games with a purpose". *Computer*, **39**(6), pp. 92–94.
- [17] Schroppe, M., 2013. "Solving tough problems with games". *Proceedings of the National Academy of Sciences*, **110**(18), pp. 7104–7106.
- [18] Abbeel, P., and Ng, A. Y., 2004. "Apprenticeship learning via inverse reinforcement learning". In Proceedings of the twenty-first international conference on Machine learning, ACM, p. 1.
- [19] Folkesson, A., Andersson, C., Alvfors, P., Alakiula, M., and Overgaard, L., 2003. "Real life testing of a hybrid pem fuel cell bus". *Journal of Power Sources*, **118**(1), pp. 349–357.
- [20] Gao, D., Jin, Z., and Lu, Q., 2008. "Energy management strategy based on fuzzy logic for a fuel cell hybrid bus". *Journal of Power Sources*, **185**(1), pp. 311–317.
- [21] Lin, C.-C., Peng, H., Grizzle, J. W., and Kang, J.-M., 2003. "Power management strategy for a parallel hybrid electric truck". *IEEE Transactions on Control Systems Technology*, **11**(6), pp. 839–849.
- [22] Evans, D. G., Polom, M. E., Poulos, S. G., Van Maanen, K. D., and Zarger, T. H., 2003. Powertrain architecture and controls integration for gm's hybrid full-size pickup truck. Tech. rep., SAE Technical Paper.
- [23] Fish, S., Savoie, T., and Vanicek, H., 2001. "Modeling hybrid electric hmwv power system performance". *IEEE Transactions on Magnetics*, **37**(1), pp. 480–484.
- [24] Antoniou, A., Komyathy, J., Bench, J., and Emadi, A., 2005. "Modeling and simulation of various hybrid electric configurations of the high-mobility multipurpose wheeled vehicle (hmwv)". In IEEE Vehicle Power and Propulsion Conference, IEEE, pp. 507–514.
- [25] Ed Bargar, H., Li, J., Goering, D. J., and Lee, J. H., 2003. "Modeling and verification of hybrid electric hmwv performance". In The 29th Annual Conference of the IEEE Industrial Electronics Society, Vol. 1, IEEE, pp. 939–944.
- [26] Liu, J., 2007. "Modeling, configuration and control optimization of power-split hybrid vehicles". PhD thesis, The University of Michigan.
- [27] Fathy, H. K., Reyer, J. A., Papalambros, P. Y., and Ulsov, A., 2001. "On the coupling between the plant and controller optimization problems". In Proceedings of the 2001 American Control Conference, Vol. 3, IEEE, pp. 1864–1869.
- [28] Peters, D., Papalambros, P., and Ulsoy, A., 2010. "Relationship between coupling and the controllability grammian in co-design problems". In Proceedings of the 2010 American Control Conference, IEEE, pp. 623–628.
- [29] Allison, J. T., Guo, T., and Han, Z., 2014. "Co-design of an active suspension using simultaneous dynamic optimization". *Journal of Mechanical Design*, **136**(8), p. 081003.
- [30] Allison, J. T., 2013. "Plant-limited co-design of an energy-efficient counterbalanced robotic manipulator". *Journal of Mechanical Design*, **135**(10), p. 101003.
- [31] Liu, J., and Peng, H., 2010. "A systematic design approach for two planetary gear split hybrid vehicles". *Vehicle System Dynamics*, **48**(11), pp. 1395–1412.
- [32] Zhang, X., Li, C.-T., Kum, D., and Peng, H., 2012. "Prius+ and volt- : Configuration analysis of power-split hybrid vehicles with a single planetary gear". *IEEE Transactions on Vehicular Technology*, **61**(8), pp. 3544–3552.
- [33] Cheong, K. L., Li, P. Y., and Chase, T. R., 2011. "Optimal design of power-split transmissions for hydraulic hybrid

- passenger vehicles”. In Proceedings of 2011 American Control Conference, IEEE, pp. 3295–3300.
- [34] Liu, J., and Peng, H., 2006. “Control optimization for a power-split hybrid vehicle”. In Proceedings of 2006 American Control Conference, IEEE.
- [35] Delprat, S., Guerra, T., and Rimaux, J., 2002. “Control strategies for hybrid vehicles: optimal control”. In Proceedings of 56th IEEE Vehicular Technology Conference, Vol. 3, IEEE, pp. 1681–1685.
- [36] Kim, N., Cha, S., and Peng, H., 2011. “Optimal control of hybrid electric vehicles based on pontryagin’s minimum principle”. *IEEE Transactions on Control Systems Technology*, **19**(5), pp. 1279–1287.
- [37] Paganelli, G., Delprat, S., Guerra, T.-M., Rimaux, J., and Santin, J.-J., 2002. “Equivalent consumption minimization strategy for parallel hybrid powertrains”. In Proceedings of 55th IEEE Vehicular Technology Conference, Vol. 4, IEEE, pp. 2076–2081.
- [38] Bellman, R., 1957. *Dynamic Programming*, 1 ed. Princeton University Press, Princeton, NJ, USA.
- [39] Abbeel, P., Coates, A., and Ng, A. Y., 2010. “Autonomous helicopter aerobatics through apprenticeship learning”. *The International Journal of Robotics Research*.
- [40] Ren, Y., and Bayrak, A. E., 2014. ecoracer. <http://ecoracer.herokuapp.com>.
- [41] Lembcke, S., and Gentle, J., 2007. Chipmunk physics js. <https://github.com/josephg/Chipmunk-js>.
- [42] Jones, D., Schonlau, M., and Welch, W., 1998. “Efficient global optimization of expensive black-box functions”. *Journal of Global Optimization*, **13**(4), pp. 455–492.
- [43] Chang, C.-C., and Lin, C.-J., 2011. “Libsvm: a library for support vector machines”. *ACM Transactions on Intelligent Systems and Technology (TIST)*, **2**(3), p. 27.
- [44] Stanley, K. O., and Miikkulainen, R., 2002. “Efficient evolution of neural network topologies”. In Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2, IEEE, pp. 1757–1762.
- [45] Konda, V. R., and Tsitsiklis, J. N., 1999. “Actor-critic algorithms.”. In NIPS, Vol. 13, pp. 1008–1014.