An Evolutionary Algorithm for Bilevel Optimisation of Men's Team Pursuit Track Cycling

Claire Diora Jordan
School of Computer Science
The University of Adelaide
Australia, SA, 5005
Email: diora.jordan@adelaide.edu.au

Trent Kroeger
School of Computer Science
The University of Adelaide
Australia, SA, 5005
Email: trent.kroeger@cs.adelaide.edu.au

***Abstract*—Evolutionary Computation is useful in a broad range of practical applications, however currently generalized algorithms tend to be focused upon solving problems in a theoretical domain. We aim to develop a range of generalised algorithms more suited than current algorithms to practical applications. We contextualize our algorithms using the elite sport of Team Pursuit Track Cycling, which features as part of the Summer Olympics. The sport is fiercely competitive and fractions of a second often separate the world's leading teams. We set about using Evolutionary Computation to optimise strategies for elite teams of cyclists through changes in the transition timings and the riders power outputs. We trial our range of Evolutionary Computation methods, comparing various algorithms and running them within a time frame suitable for use in a real world environment. We find significantly better results are able to be obtained through our methods than current strategies being developed at an elite level and find the use of the developed algorithms favourable for use in a practical environment.**

I. Introduction

Evolutionary Algorithms can be applied to a broad range of practical problems. We present a series of general evolutionary algorithms and use these to optimize a strategy for cyclists competing at an elite level track-based cycling event, namely the Men's Team Pursuit, in order to provide decision support to coaches and athletes. This track based cycling event presents a particularly challenging problem that involves hierarchical dual level solutions spread over a multi-modal solution space. Due to the practical nature of our problem we require algorithms that are able to be run in a computationally feasible length of time, namely a few hours, so that they can be run on field prior to a race.

Decision support systems that use artificial intelligence have previously been used in various sport fields. Neural networks have been used to find optimum strategies in sport biomechanics, resulting in optimisation of cricket fast bowling techniques, javelin throws and soccer kicking [1]. Additionally evolutionary algorithms have been used to help analyse various aspects of different sports, such as to set the final innings target for cricket [2], and to develop strategies, such as speed and racing line for simulated car racing [3] [4].

Elite cycling has previously been modelled mathematically [5], however this has been done so for a single cyclist, where an all-out pacing strategy is likely to be optimal. The addition of extra cyclists to the problem in a team-based event means that the solution requires sub-maximal efforts throughout the race, hence the current mathematical models are unable to be used in our scenario. We also have several constraints that arise specific to track based cycling, primarily that a team of riders who are riding in unison do so with a lead rider who heads the team followed by the remainder of the team's riders who benefit from the lead rider's slipstream. This slip-streaming effect is an important consideration as it allows riders to conserve energy, as the lead rider typically uses a greater amount of energy. Teams apply a transition strategy, where the lead rider is changed for another, so as that the riders may each expend their energy evenly. The problem is further complicated by considering a men's cycling team. The men's event allows for one of the individual riders to remove themselves from the race, only requiring that three out of the four riders complete the entirety of the race.

Our problem is formulated with two distinct sets of variables to be optimised. We are able to optimise both the riders transition strategy, i.e. when during the race a new rider is transitioned to lead, and their power strategy, i.e. how much power, or energy, the lead rider is required to exert in order to ride at the necessary pace. The transition strategy of the riders is co-constrained by the pacing strategy and an optimal transition strategy for a single pacing strategy may not be optimal for all pacing strategies. Implicitly, the minimization of overall time using the pacing strategy is dependant on having an optimal transition strategy. This dual level optimisation presents a complex set of solutions as for any given transition strategy it may not be possible to find an optimal pacing strategy and hence sub-optimal solutions for the overall problem may be found.

Current literature surveyed shows a limited ability to deal with this type of dual level problem, with the majority of current algorithms becoming stuck in local optima [6]. Further, most current algorithms deal with theoretical problems and do not map well to many problems that are presented within a practical setting. Our problem space is multi-modal and complex and hence local optima present a real problem for algorithms used. Many current algorithms rely upon explicit knowledge of the search space, however in our real life application, such information is not available.

We present a generalised and extensible approach to solving

these dual level style problems, and apply them to the real world problem space presented by the men's team pursuit cycling event. We present algorithms that are able to be run within a fixed number of evaluations and hence be computationally feasible in a practical sense. This concrete example shows a proof of concept for a challenging problem and it is hoped that the algorithms would be adaptable to a range of practical based problems.

II. Problem Description

Team Pursuit Track cycling involves a team of riders competing in a timed event around a banked elliptical circuit known as a velodrome. [7] presented a framework for modelling the women's team pursuit problem that consists of three riders in a team. We present here a framework for a men's team, which has a major point of difference with there being four riders in the event. Of these four riders only three are required to finish the entirety of the race, with one rider being able to drop-out before the finish. This differs from the women's case where all riders are required to complete the entire distance of the event. The men's event is also competed over a longer distance of 4000 meters, consisting of 16 laps of the velodrome, compared with the women's which is run over 3000 meters (12 laps). This makes the men's case significantly more complex and enlarges the search space dramatically.

The framework must also take into account constraints that are common across both the men's and women's cases, and overall the representation for both cases is similar. It must be considered that the lead rider at any given time must exert more energy than the following riders due to the effects of slipstreaming. This dynamic requires the development of a strategy that allows the riders to rotate, or *transition* between positions. These transitions generally occur on the banked sections of the velodrome, i.e. either of the two turns, as this is where the cyclists are able to most efficiently sweep up the turn and change their relative position. All transitions move the lead rider to the rear position, elevating the first following rider to the lead position. The ordering of the riders never changes during the event, only their relative positioning. Teams have their own preferred ordering for the riders.

We also find the *power* which the riders will need to use on each half lap of the race. This power is the rate of energy being exerted by the lead ride, measured in Watts. The energy being used by the following riders is generally less, due to the aforementioned slipstreaming effect, and it is derived based upon keeping up with the lead riders acceleration.

III. Problem Formulation

In the previous section we determine that the core parameters for a cycling strategy are:

- The number of half laps the lead rider performs before transitioning - this is a discrete variable and thus can be represented by an integer.
- The power output per half lap of the first rider, otherwise known as the pacing strategy - this is measured in Watts and thus must be represented as a float to capture its continuous nature.

In order to maintain simplicity, we fix the number of variables representing the strategy. This is equal to the number of effective half laps of the race to represent the transition strategy, in case the riders transition every half lap. If the riders transition less often, the extraneous variables are simply ignored. Similarly to represent the pacing strategy the number of variables is the number of effective half laps, with each variable representing the power the lead rider exerts for that half lap. Unlike the transition strategy, the amount of power levels is fixed to the amount of half laps, and there are no extraneous variables. This representation can be seen in Figure 1. It should be noted that the number of effective half laps is the number of laps the rider can transition. The riders cannot transition for the first or last 0.75 laps, hence these are represented as a single half laps in the genome.

Fig. 1. Illustration of the genome representation. HL denotes the half-laps before rider transition and P indicates power output of the lead rider per half-lap.

A key part to the genome representation is the simplicity of representing the solution strategy. Although the men's team pursuit event has the additional complexity of having a rider that does not have to complete the race, it was decided that this would not be represented in the genome, but rather be modelled in the fitness evaluation function. This creates a generalised genome that standard Evolutionary Computation operators can then be applied to.

IV. Evaluation Function

The aim of the optimisation is to minimize time, hence the fitness evaluation function uses time as a measure of fitness. The fitness function also needs to assure that the riders only expend the amount of energy they have available for the race. If one rider runs out of energy they are assumed to have dropped out, however if two riders run out of energy the solution becomes infeasible and a penalty function is applied. The algorithm can be seen in Figure 2.

In order to obtain the time taken to ride a half lap, a forward integration technique is used, which breaks the time into small Δt sections of 0.1 seconds. The acceleration for riding this Δt is calculated using Equation 1:

$$\Delta KE = (P \times E - C_D A \times (\frac{1}{2}\rho \mathrm{v}^3) - \mu \times (\mathrm{v} F_N)) \times \Delta t \quad (1)$$

where KE is kinetic energy, P is power, E is mechanical efficiency, $C_D A$ is the frontal area of the bike and rider, ρ is

Fig. 2. Fitness Function

Input: list of transitions $transitions$, list of power levels $powers$
Output: predicted race time $totalTime$

$totalTime := 0$
$transCount := 0$ {where in the transition strategy we are up to}
$halfLapsLeft := transitions[transcount]$ {number of half laps left before the next transition}
for $i \in \{1, \ldots, powers.length\}$ **do**
  $p := powers\,[i]$
  $t := forwardIntegration(\ldots)$ {compute the time needed for cycling a half lap, using Equation 1}
  $totalTime \mathrel{+}= t$
  $rider_1.totalEnergy \mathrel{-}= p \cdot t$
  Reduce the following riders' energies using Equation 2, based on them keeping up with $rider_1$'s speed
  $halfLapsLeft --$ {1 less half lap until transition}
  if $(halfLapsLeft) = 0$ **then** {time to transition}
    reorder the riders
    $transCount\ ++$
    $halfLapsLeft := transitions[transcount]$ {get the number of half laps left before the next transition}
  if not at least 3 riders completing the race **then**
    $totalTime := 1000$ {apply a penalty}

air density, v is speed, μ is a global coefficient of friction, F_N is the weight of the bike and the rider, and t is time.

The energy needed for this acceleration is subtracted from the lead riders energy. The energy for the following riders to keep up with the leader rider for time Δt is found using Equation 2:

$$P = (C_D A \times C_{Draft} \times \frac{1}{2}\rho \mathrm{v}^3 + \mu \times (\mathrm{v}F_N) + \frac{\Delta KE}{\Delta t})/E \quad (2)$$

where E is the efficiency of the drive system, ΔKE is calculated using the final velocity of the first rider after acceleration. C_{Draft} is drafting coefficient of the rider and represents the reduction in the $C_D A$ of a cyclist due to the aerodynamic benefits of drafting in second, third or fourth position.

This energy is then subsequently subtracted from their total energies. This process is then repeated until the distance of a half lap is covered, and the sum of the Δts is the time taken to complete this half lap.

These equations were derived from [5], and adapted to suit team pursuit track cycling conditions by removing potential energy (as vertical elevation is negligible) and combining air and ground speed (as the event is run in still air conditions) in the case of Equation 1, and adding a draft benefit (C_{Draft}) from being in the slipstream of the other riders in the case of Equation 2.

Fig. 3. Single Level Evolutionary Algorithm

for $i \in \{1, \ldots, population.size\}$ **do**
  Choose initial x uniformly at random
  $population.add(x)$
repeat
  for $i \in \{1, \ldots, children.size\}$ **do**
    Choose parent x using tournament selection
    Create x^* by mutating each position of x using the appropriate pacing or transition mutator for the given position
    $children.add(x^*)$
  Combine the parent and children populations
  Replace $population$ with the fastest $population.size$ solutions
until reach max number of fitness evaluations
return $fastest(population)$

V. Algorithm Design

A. Single Level Optimisation

Initial experiments involved using a non-problem specific optimisation technique, that does not take into account the dual nature of the problem. That is, it does not account for the fact that there are two subproblems present in the problem (that of optimising the transition strategy and optimising the power strategy), and rather mutates the solution as one large problem. This can be done because our generalised representation of the problem allows for standard Evolutionary Computation techniques to still be applicable. The algorithm is outlined in Figure 3, and the operators used are explained in greater detail in the following.

Due to the fact that the first half of the problem, the transition strategy, is represented as discrete variables, and the second half of the problem, the power strategy, as continuous variables, different mutation methods have to be used for each half of the problem. Two alternative standard mutation methods are proposed, as outlined in [8]. For the discrete transition strategy:

- Random Mutation: First the effective length m of the existing strategy (i.e. the subsequence of transitions required to finish the race) is calculated, with extraneous variables being ignored. An iteration over the discrete part of the genome is then performed, with probability $p = 1/m$ of a new random integer being chosen for each given position.
- Creep Mutation: This type of mutation uses stepwise mutation to only permit small changes. First the effective length m of the existing strategy is calculated, with extraneous variables being ignored. An iteration over the discrete part of the genome is then performed, with probability $p = 1/m$ for each given position of the integer 'creeping' either up or down by 1.

For the continuous pacing strategy:

- Uniform Mutation: An iteration is performed over the continuous part of the genome, with probability $p = 1/n$, of the float at that position being replaced, where n is

the size of the genome. The value of the chosen gene is replaced with a uniform random value selected between the upper and lower bounds for the power.

- Non-Uniform Mutation: This type of mutation decreases the amount of mutation as the generation number increases. This keeps the population from stagnating in the early stages of evolution, but allows the solution to be refined in the later stages of evolution. This works in the same way as the uniform mutation, however rather than replacing the float value, a value chosen using a Gaussian Distribution that narrows as the number of generations increases, is added or subtracted from the existing value, with the new value being clipped if it falls outside the upper or lower bound for the power.

Due to the problem nature the pacing and transition strategy are highly interrelated, making crossover difficult, It would only make sense to perform crossover for a pacing strategy when the transitions are the same, and there is no way to ensure that two parents would have the same transition strategy. As such no crossover was used.

B. Bi-Level Optimisation

The team pursuit track cycling problem can be viewed as a type of Bi-Level optimisation problem (BLP) [9]. It has a hierarchy of two levels of decision makers, with two parts to the problem, a leader and a follower. In a BLP problem the decision making at the leader level influences the decision making at the follower level. We have applied the men's team pursuit problem as a BLP, with the transition strategy as the leader, and the pacing strategy as the follower. The transition strategy is chosen as the leader as it has a smaller search space then the pacing strategy, allowing for a larger proportion of this search space to be explored.

Specifically, the given problem is an example of a parametric BLP, where minimising the follower (pacing) strategy is implicitly contained in the minimization of the leader (transition) strategy, which means that in order to get the minimum time for a given transition strategy, the pacing strategy also needs to be optimised. Parametric BLPs are widely considered to be more complicated than other BLPs [10].

The algorithm splits the problem into two separate problems, the leader (transition) problem and follower (pacing) problem, and a separate EA is used for each of the two problems. The leader problem then calls the EA for the follower problem each time a new solution for the leader problem is found, in order to obtain the matching optimised follower problem. The results of the leader and optimised follower problem are then combined and used as the parents for the next generation of the leader problem. The algorithm is outlined in Figure 4. Of particular note is the fact that the inner algorithm is allowed to converge on an optimal solution, that is it is run until no improvement in time is made for a number of evaluations, which is set to 1000. Unlike using a fixed number of evaluations, this allows for the minimisation of the follower problem contained within the leader problem.

Fig. 4. Bi-Level Evolutionary Algorithm

for $i \in \{1, \ldots, population.size\}$ **do**
  Choose initial x for transition strategy uniformly at random
  Choose initial y for pacing strategy uniformly at random
  $transPopulation.add(x, y)$
  $pacingPopulation.add(x, y)$
repeat
  for $i \in \{1, \ldots, transChildren.size\}$ **do**
    Choose parent $P_{x,y}$ using tournament selection
    Create x^* using the transition mutator on x
    repeat
      for $i \in \{1, \ldots, pacingChildren.size\}$ **do**
        Create y^* by using the pacing mutator on y
        $pacingChildren.add(x^*, y^*)$
      Combine the parent and children pacing populations
      Replace $pacingPopulation$ with the fastest $pacingPopulation.size$ solutions
    until no time improvement for n evaluations
    $transChildren.add(fastest(pacingPopulation))$
  Combine the parent and children transition populations
  Replace $transPopulation$ with the fastest $transPopulation.size$ solutions
until reach max number of fitness evaluations for transition strategy
return $fastest(transPopulation)$

C. Co-Evolutionary Optimisation

In a survey of current literature it has been found that current techniques are limited in their abilities to deal with parametric BLP problems, and the majority get stuck in local optima rather than finding the global optima [9]. Due to the fact that this type of problem is non-convex and presents a complex search space, in order to overcome this many current algorithms rely on knowledge of the search space. Hence many existing algorithms are not suitable for application to real life problems, in which it is difficult to obtain such knowledge [6].

The generalised BLP algorithm presented in the previous section does this, using knowledge of the fact the transition strategy has a smaller search space to use it as the leader problem, thus reducing the overall search space of the problem. We attempt to overcome this by developing an algorithm that balances between the two parts of the problem, so that rather than having a leader and follower problem, the two parts of the problem are coevolved. In this way there is symmetric cooperation between the two sections of the problem, combining the best results of each section of the problem to get the best overall result, rather than finding the best follower for a given leader.

The algorithm works by having an island for each part of the problem, which works on one section of the problem. One island contains a fixed population of x variables, which are the solution for the other part of the problem (in this case the

pacing strategy), while optimizing the y variables (the transition strategy). The other island contains a fixed population of y variables (the transition strategy), while optimising for the x variables (the pacing strategy). Every 1000 generations a combination operator is called, which takes the best half of one islands population and replaces the worst half of the other islands population. In this way it shares the best solutions for each part of the problem amongst both the islands so that they can be used in future generations, thus pushing the solutions to the optimal solution for the combination of the parts. The algorithm can be seen in Figure 5.

It should be noted that this type of algorithm, unlike the standard Bi-Level approach, has the ability to be parallelized, so that each island can be run on a different computer, thus enabling the problem to be solved in a faster time. Additionally if given a multilevel problem with more dimensions than two, more islands can be created that work in parallel. Contrary to this, the BLP solution would have to have additional sublevels of optimisation, which dilutes the amount of solutions that can be evaluated for the leader part of the problem.

Fig. 5. Co-Evolutionary Algorithm

for $i \in \{1, \ldots, population.size\}$ **do**
  Choose initial x for transition strategy uniformly at random
  Choose initial y for pacing strategy uniformly at random
  $transPopulation.add(x, y)$
  $pacingPopulation.add(x, y)$
repeat
  for $i \in \{1, \ldots, transChildren.size\}$ **do** {Island 1}
    Choose parent $P_{x,y}$ using random selection on transPopulation
    Create x^* using the transition mutator on x
    $transChildren.add(x^*, y)$
    Combine the parent and children transition populations
    Replace $transitionPopulation$ with the fastest $transPopulation.size$ solutions
  for $i \in \{1, \ldots, pacingChildren.size\}$ **do** {Island 2}
    Choose parent $P_{x,y}$ using random selection on pacingPopulation
    Create y^* using the pacing mutator on y
    $pacingChildren.add(x, y^*)$
    Combine the parent and children pacing populations
    Replace $pacingPopulation$ with the fastest $pacingPopulation.size$ solutions
  if generation % 1000 = 0 **then** {Combination Operator}
    transPopulation := fastest half of *transPopulation* + fastest half of *pacingPopulation*
    pacingPopulation := fastest half of *pacingPopulation* + fastest half of *transPopulation*
  $bestSolution = fastest(transPopulation, pacingPopulation, bestSolution)$
until reach max number of fitness evaluations
return $bestSolution$

VI. Experimental Design

In order to compare the proposed operators, experiments were performed on the Single Level algorithm, using a population of size 1 and combinations of the various operators for the discrete transition strategy and continuous pacing strategy in order to find the best operators. This was done using 2 million evaluations (as this was found to be the number of evaluations required for the algorithm to converge on a solution), and 30 repetitions of each experiment were performed. The best combination of pacing and transition mutators were then used for all further experiments.

In order to compare algorithms, a fixed number of evaluations was used. That is, each algorithm is allowed to call the fitness evaluation function only a certain number of times. This is due to the fact that each algorithm works differently (with the Bi-Level algorithm calling an algorithm within an algorithm and the Co-Evolutionary algorithm running two algorithms in parallel), and it would be difficult to run them for an equivalent amount of generations to create a fair comparison.

As with the operator experiments, the number of evaluations was set at 2 million, and 30 repetitions of each experiment were performed. Although preliminary testing showed that the Bi-Level and Co-Evolutionary algorithms were able to obtain better solutions with more evaluations (the results of which can be seen in Table IV), 2 million evaluations takes about three hours to run on a 2.4GHz Intel Core 2 Duo, and more than this would be considered impractical for use on the field prior to a real cycling event. This is due to the fact that the problem has real world properties that change and have influence on a race, such as the temperature and humidity.

Given the limit on the number of evaluations, different sized populations would change the result outcome. A smaller population would not hold as many seed parent solutions, however would be able to run for more generations than a larger population. Thus in order to observe the effects of population size on the algorithms, all three aforementioned algorithms were run with population sizes of 1, 10 and 100. Note due to the nature of the Co-Evolutionary algorithm, it requires a minimum population of 2 (one for each island), however this difference is considered negligible for comparison purposes.

To be able to create a realistic model of the race, we have implemented the problem with the parameters shown in Table I. Of particular interest is that we approximate the power a rider can output by their mass multiplied by 6 Wkg^{-1} (the mean maximum power used by an elite male cyclist), multiplied y 240 seconds. We impose a bound between $200W$ and $1300W$ of energy to be used per half lap, to ensure a realistic amount of energy is used, and model fatigue by limiting the number of half laps a rider can be in the lead position to 3.

VII. Experimental Results

The results of using the combinations of the various operators for the transition and pacing strategy on the Single Level 1+1 EA algorithm are provided in Table II. As can be

TABLE I
PROBLEM PARAMETERS

Mechanical efficiency	0.977
Global friction	0.0025
Temperature	20° C
Air pressure	1013.25 hPa
Relative humidity efficiency	50%
Race distance	4000 m
Maximum power	1300 W
Minimum power	200 W
Maximum half-laps before transition	3
Effective Half Laps	30
Rider mass	75.0 kg
Rider height	1.75 m
Bicycle mass	7.7 kg
Rider mean maximum power	6 W
C_{Draft} second position	0.75
C_{Draft} third position	0.65
C_{Draft} fourth position	0.55
Time to transition	0.12 s

seen using Random Mutation for the transition strategy and Non-Uniform Mutation for the pacing strategy provides the best results, and hence this combination is used for all further experiments.

TABLE II
OPERATOR RESULTS

	Uniform Random	**Uniform Creep**	**NonUniform Random**	**NonUniform Creep**
avg	254.1	254.3	252.5	252.6
stdev	2.8	2.5	2.3	2.9

In order to form a basis for camparison with the two level optimised solutions, we investigate a set transition strategy, where the first rider performs that initial 0.75 laps, and then the riders transition every two laps (a standard strategy used in competitive team pursuit). The pacing strategy for this was then optimised using the Single Level 1+1 EA with no transition mutator. The results of our algorithms are summarised in Table III. As can be seen, using our optimised algorithms we were able to achieve favourable improvements in race time.

TABLE III
RESULTING TIME OF OPTIMISED SOLUTIONS USING 2 MILLION EVALUATIONS

Algorithm	**Best Race Time (s)**	**Average Race Time (s)**	**Standard Deviation**
Unoptimised Transition	250.99	252.54	0.97
Single Level 1+1	249.62	252.52	2.36
Single Level Pop 10	246.82	249.99	1.65
Single Level Pop 100	246.62	250.00	1.73
Bi-Level 1+1	247.02	249.02	1.35
Bi-Level Pop 10	247.22	248.98	0.70
Bi-Level Pop 100	248.22	250.18	1.31
Co-Evolutionary Pop 2	247.82	250.93	1.90
Co-Evolutionary Pop 10	246.82	249.86	1.80
Co-Evolutionary Pop 100	247.62	250.93	1.70

Our Bi-Level algorithm was able to achieve the best average results, however the majority of algorithms were able to find a minimal result close to 247. As the algorithms are evolutionary this can be expected, however the low standard deviation shows stability in our algorithms, especially the Bi-Level.

TABLE IV
RESULTING TIME OF OPTIMISED SOLUTIONS USING 6 MILLION EVALUATIONS

Algorithm	**Best Race Time (s)**	**Average Race Time (s)**	**Standard Deviation**
Unoptimised Transition	250.99	252.54	1.04
Single Level 1+1	248.52	252.51	1.93
Single Level Pop 10	246.82	249.97	1.86
Single Level Pop 100	246.62	249.98	1.43
Bi-Level 1+1	247.02	248.97	1.22
Bi-Level Pop 10	247.11	247.96	0.81
Bi-Level Pop 100	247.98	249.99	1.20
Co-Evolutionary Pop 2	247.02	249.93	1.92
Co-Evolutionary Pop 10	246.12	248.65	1.70
Co-Evolutionary Pop 100	247.62	249.80	1.55

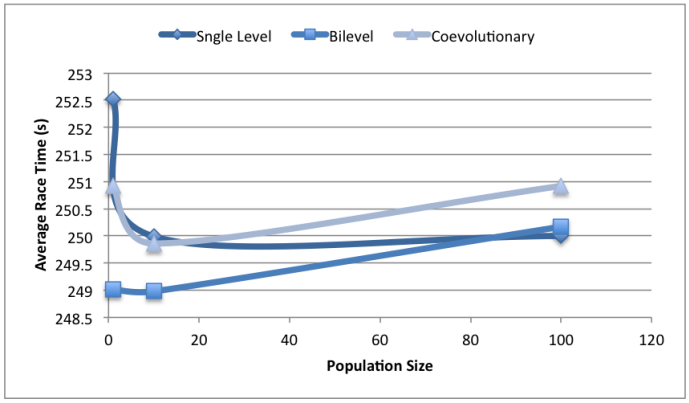

Fig. 6. Comparison of population size effect on the Single Level, Bi-level and Co-Evolutionary algorithms using 2 million evaluations

We present our results graphically in Figure 6 in order to be able to see which algorithm is best and also to be able to analyse the effect of population size on the algorithms.

As can be seen in Figure 6 a population of size 10 is best for all of the algorithms.

VIII. ANALYSIS AND DISCUSSION

It was observed in the previous section that the use of Bi-Level optimisation is best in this scenario, however we will analyse if it is significantly better. As the results cannot be assumed to be normally distributed, we use an unpaired, non-parametric statistical hypothesis test, the Mann-Whitney-Wilcoxon test, to compare the results of our algorithms.

We combined the results of all population sizes for comparison to provide a larger pool of results to analyse, and as such we also make no inference about which population size is best. These results provide an analysis of which algorithm is able to perform best within our set number of iterations, regardless of population size.

Table V shows us that the Bi-Level algorithm is indeed significantly superior to the Co-Evolutionary algorithm. In turn the Co-Evolutionary algorithm outperforms the Single Level

TABLE V
OVERALL MANN-WHITNEY-WILCOXON COMPARISON ON THE UNOPTIMISED, SINGLE LEVEL, BI-LEVEL AND CO-EVOLUTIONARY ALGORITHMS

Alg A	Alg B	Mean$_A$	Mean$_B$	U$_A$	*p*	Sig
Single Level	Unoptimised	251.35	252.54	709	0.015	+
Coev	Single Level	250.29	251.35	4077.5	0.0122	+
Bi-Level	Coev	249.39	250.29	5658	0.0011	+

optimisation, and all algorithms outperform the unoptimised transition strategy.

We conjecture that the Bi-Level is able to work best in this scenario as it takes the nature of the search space into account. Consider that the number of transition strategies is 2^{32}, as there are 32 possible transition points and for each is either a binary transition or stay. The size of the search space for the power strategy is near infinite. As such, treating the transition as the leader in this space and the power strategy as the follower is optimal. The Co-Evolutionary strategy would most likely be better suited to a problem type where each of the levels had a similarly sized search space, as we would be able to traverse them equally. We also would conjecture that the Co-Evolutionary strategy is more extensible, and could be moved into a Multi-Evolutionary strategy. This potential makes the Co-Evolutionary strategy far more extensible for both other practical applications as well as possible extensions within our own case.

We now will analyse the population sizes to explore which size is able to achieve the best results. We look at the various population sizes across all three algorithms and find if there is a statistically better population size to be used. From our initial results we believe the order of suitability for our population size to be 10, 100, 1 and hence will analyse them in this order.

TABLE VI
MANN-WHITNEY-WILCOXON COMPARISON ON POPULATION RESULT OF EACH ALGORITHM

Pop A	Pop B	Mean$_A$	Mean$_B$	U$_A$	*p*	Sig
Bi-Level						
10	100	248.98	250.17	699.5	0.0001	+
100	1	250.17	249.02	248	0.0014	+
Co-Evolutionary						
10	100	249.86	250.92	622.5	0.0055	+
100	1	250.92	250.93	459	0.4483	-
Single Level						
10	100	249.99	250.00	440.5	0.4483	-
100	1	250.00	252.52	242	0.0021	+

As we can see from Table VI, for both the Bi-Level and Co-Evolutionary a population size of ten is indeed best. For Single Level this is not the case, however it is slightly better than 100, and significantly better than 1. As such we believe a population size of 10 is the most suitable.

We believe this is a factor of our evolutionary method which computes to a set number of calls to the fitness function. By having a large population size, 100 evaluations need to be made every generation, and hence the number of generations is small, leading to small room for improvement. Indeed this is somewhat of an imposed ceiling, as generally a large population size is superior, however due to the practical setting of our problem we must impose that the algorithm is able to be run in a computationally sensitive amount of time. This is due to the fact that in a practical scenario these algorithms may need to be run quickly as the physical nature of the event may change. Strategies may need to be significantly changed for events such as a rider being injured on a given day, or for physical occurrences, such as a change in relative humidity. The investigation of finding the most "optimal" strategy is left as future work, and our algorithm is only considered in a time frame suitable in the scenario for which it is being run.

IX. CONCLUSION AND FUTURE WORK

In this paper we addressed the issue of a real life dual part optimisation problem, evolving strategies for Men's Team Pursuit Track Cycling. We introduced a model that allowed us to simulate and evaluate strategies, and several algorithms that address the nature of a dual part optimisation problem, which attempt to overcome the issues of developing and applying generalised algorithms to real life problems.

The experiments revealed several interesting insights. Even the simplest of algorithms, the Single Level optimisation, which did not take into account the dual nature of the problem, was able to provide better solutions than the unoptimised solutions for the domain. However it was found that the Bi-Level optimisation, that takes into account more knowledge of the solution space, did provide significantly better results than did the generalised algorithms.

Of specific interest is the fact that the use of generalised algorithms proved sufficient for use in our domain, and may be more widely applicable to other real world applications.

In the future we will continue research in the following areas:

- We are currently in the process of calibrating these results with professional cyclists.
- Application of these algorithms to other real life parametric bilevel problem domains.
- Further investigation of the effects of population size on algorithm performance.
- The development of the presented algorithms to work with multi-objective problems, and the application of this to elite tack cycling events, where both the time taken and energy used by each cyclist is a consideration.
- The production of a stochastic model that gives the sensitivity range of a given solution to parameter changes. As there may be error in parameter measurement, an extra objective would then be to reduce the standard deviation of the stochastic model's distribution of results, to try and get a solution with a fast race time that is also less sensitive to parameter changes.

ACKNOWLEDGMENT

The authors would like to thank Dr David Martin from the Australian Institute of Sport Cycling Program and Dr Tammie Ebert from Cycling Australia for their valuable support.

REFERENCES

[1] R. Bartlett, "Artificial intelligence in sports biomechanics: New dawn or false hope?" *Journal of Sports Science and Medicine*, vol. 5, no. 443, pp. 474–479, 2006.

[2] P. Scarf and X. Shi, "Modelling match outcomes and decision support for setting a final innings target in test cricket," *IMA Journal of Management Mathematics*, vol. 16, no. 2, p. 161, 2005.

[3] J. Togelius and S. Lucas, "Evolving controllers for simulated car racing," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2. IEEE, 2005, pp. 1906–1913.

[4] L. Cardamone, D. Loiacono, and P. Lanzi, "On-line neuroevolution applied to the open racing car simulator," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 2622–2629.

[5] J. C. Martin, A. S. Gardner, M. Barras, and D. T. Martin, "Modeling sprint cycling using field-derived parameters and forward integration," *Medicine and Science in Sports and Exercise*, vol. 38, no. 3, pp. 592–597, 2006. [Online]. Available: http://www.biomedsearch.com/nih/Modeling-sprint-cycling-using-field/16540850.html

[6] R. Mathieu, L. Pittard, and G. Anandalingam, "Genetic algorithm based approach to bi-level linear programming," *RAIRO. Recherche opérationnelle*, vol. 28, no. 1, pp. 1–21, 1994.

[7] M. Wagner, J. Day, D. Jordan, T. Kroeger, and F. Neumann, "Evolving pacing strategies for team pursuit track cycling," in *9th Metaheuristics International Conference*, 2011.

[8] A. Eiben and J. Smith, *Introduction to evolutionary computing*. Springer Verlag, 2003.

[9] U. Wen and S. Hsu, "Linear bi-level programming problems–a review," *Journal of the Operational Research Society*, pp. 125–133, 1991.

[10] V. Oduguwa and R. Roy, "Bi-level optimisation using genetic algorithm," in *Artificial Intelligence Systems, 2002.(ICAIS 2002). 2002 IEEE International Conference on*. IEEE, 2002, pp. 322–327.