On BURP investments

http://cbeaume.com/en/teaching_f1.html

c.m.l.beaume@leeds.ac.uk

December 24, 2021

1 Problem statement

In the F1 Strategy Competition, we are given an overall budget to split between 4 areas: (i) Marketing boosts chances of signing the best drivers, (ii) Reliability decreases the risk of mechanical failure during the races, and (iii) Chassis and (iv) Engine increase the performance of the car. By construction, Marketing is a key component to the budget as it is the only area where investing does not turn directly into performance but, rather, into a chance of gaining performance. In other words, certain teams will gain more than other in proportion to their investment. The rest of the budget needs to be decided according to the level of risk one wants to take: a higher Reliability implies less budget left for performance-related areas (Chassis and Engine) but makes it more likely for the car to finish the race ("if you want to finish first: first, you have to finish"). Finally, given a satisfactory Reliability investment, the final decision is that of the split between Chassis and Engine. This split can be done in multiple ways and is driven by the characteristics of the circuits used for the championship (they do not favour Chassis and Engine similarly) and by the profile of the points awarded at the end of each race.

While the above reasoning is a good initial analysis, the problem of finding how to best invest our budget is, in fact, more complex. A more rigorous analysis shows that there is no best answer in general: the best budget split depends rather heavily on the budget splits of the other teams (see Example below).

Example: Dependence of the optimal budget split on the budget split of the other teams

Let us consider two teams (A and B) fighting for two drivers ($5\star$ and $2\star$).

If both teams invest $0.0€$ in Marketing, they have both the same chance to get the best driver. If Team A decides to invest $0.1€$ in Marketing (against $0.0€$ for Team B), it will sign the best driver. Team B can prevent this by investing $0.1€$ as well into Marketing, but that would be too conservative: a Marketing investment of $0.2€$ would give them a better probability to attract the best driver than team A at a small cost.

The above reasoning works for a while but not indefinitely. To understand why, we need to remember that it is not the driver that matters but the expected team performance level (which involves Chassis and Engine) or, equivalently, its *return on investment* (ROI) on the Marketing stage. The latter is simply the difference between the driver ability and the Marketing investment[ab].

For example, let us imagine that Team B knows that Team A will invest $3.1€$ into Marketing. What should Team B do to get the upper hand? It can increase its Marketing investment to $3.2€$ and will have a slight probabilistic edge over Team A or not invest at all in Marketing. If it chooses the latter, it would get the worst driver and a ROI of $2 - 0 = 2€$ against Team A's $5 - 3.1 = 1.9\star$.

Consequently, there are two ways to exploit sufficiently large Marketing investments from other teams: investing slightly more into Marketing, and investing very little to nothing into it. In fact, the larger the Marketing investment from the competition, the more powerful withdraw one's Marketing investment is.

[a] leaving Reliability aside for the sake of simplicity

[b] this is a consequence of the fact that Chassis and Engine investments turn into performance at the same rate

In addition to the necessity of being adaptive, we possess no knowledge about the budget split of the other competitors. As a consequence, the problem of deciding how to invest our budget is an *incomplete information game*.

2 Decision-making program

2.1 BURP's objective

Given the complexity of the game, I decided to design a computational method. First, I need an objective. Although I aim to win the Constructor's Championship, finishing second is not a terrible result. I decided to give a value to each Constructor's Championship finishing position that reflects my personal satisfaction level and refer to this quantity, which I try to optimize, as the *metric* (see example below for what this metric was in Season 4).

> Season 4: BURP's metric
>
> With 14 teams participating to Season 4, I decided to value the Constructor's Championship finishing positions as follows:
> - 1^{st}: 10pts
> - 2^{nd}: 6pts
> - 3^{rd}: 4pts
> - 4^{th}: 2pts
> - 5^{th}: 1pt
> - 6^{th} to 14^{th}: 0pt
>
> This is what I refer to as the metric. My objective is to choose the budget split that maximizes the expected value of that metric.

The problem for my decision-making program is now posed as an optimization problem: what budget split maximizes the metric?

2.2 Season simulation subroutine

Before delving into the technical details of the algorithm, we need to be able to simulate seasons in order to determine Constructor's Championship rankings to calculate the above metric. I created a subroutine for that, which, given a set of budget splits, simulates a season and returns the final ranking, as well as some other data that can be used for post-treatment purposes. Unfortunately, modeling a season involves uncertainty at several levels: (i) driver allocation is carried out using a random process that cannot be ignored and (ii) the races involve a large number of random processes, which, fortunately, can be ignored[1] in such a way that the result of a race can simply be determined from the expected performance of all the participating drivers on the relevant circuit.

My subroutine proceeds in several steps. First, it allocates the drivers to the teams based on their marketing investment and the probability law in use for that season[2]. Then, knowing the full team lineups (Reliability, Chassis, Engine, Driver 1 and Driver 2), the subroutine ranks the drivers for each circuit as a function of their performance level, taking into account circuit-specific bonuses but ignoring random effects. A DNF filter is then applied for each race with probability in line with the Reliability investment of the associated team and the reliability formula associated with the competition. For each race, the drivers that did not get filtered out receive points as a function of their rank. The point attribution and the race characteristics are those specified in the championship rules.

In all the cases where this subroutine is used, we are not interested in simulating one season but, rather, in knowing the expected result for this season. To approximate the expected result, I loop over

[1] as a first-order approximation

[2] it is the same bit of code that I use to run the "official" driver allocation

my subroutine a large number of times and calculate the average result such as the averaged metric or average number of Constructor's Championship points.

2.3 Iterative method

Now that we are able to calculate the expected result for a season, we need to think of a way to tackle the optimization problem formulated in Section 2.1 by developing a way to improve budget splits. I created an iterative method to take care of it.

After a season is simulated using the subroutine explained in Section 2.2, I assume that each team possesses the knowledge of the other teams' budget split during that season and is allowed to *respond* by changing their own budget split. One key aspect of my method is to make sure that all the teams respond *simultaneously*, so that the incomplete information nature of the game is preserved. Of course, I want the teams to respond in a clever way, to make sure that my decision-making program is evolving in the right direction. I chose to handle this optimal response problem using a simple genetic algorithm for each of the teams. Note that, to model the independent nature of the response problem of a given team, it is important that each team is given its own "genetic" population and that each of the resulting genetic algorithms be treated independently from one another. Once the best response is determined for all the teams, it replaces their budget split and a new season is simulated.

The above process (season simulation, then best response) is repeated and the top-ranking budget split, according to the desired metric, is tracked after each season simulation. In case the iterative method does not converge to a well-defined budget split, iterations are carried out long enough to obtain a converged statistical description of the top-ranking budget splits. Due to the inherent stochastic nature of the method, I reinitialized and restarted my iterative method several times to ensure that the converged budget split is unique or that the statistical description provides a robust best budget split. I then use the result as BURP's budget split.

2.4 Genetic algorithm

The optimal response problem of the iterative method is probably the most tricky to set up. The idea is, however, simple: knowing the budget split of the other teams, what is the budget split I should use to maximize the metric? To answer this question, I decided to use a genetic algorithm for two reasons: (i) it is fun to play with and (ii) it does *not always* converge to the global optimum[3]. I explain below the basic principles of my simple genetic algorithm.

First of all, and before proceeding to the generation loop, I initialize the population for the genetic algorithm. Each individual is modeled using 4 genes: Marketing, Reliability, Chassis and Engine that I set randomly and whose sum satisfies the overall budget constraint[4]. I select one of these individuals randomly use it as the budget split of the team.

2.4.1 Fitness calculation

To assess the fitness of each of the individuals in the genetic population, I set up a competition lineup consisting of the budget split of all the other teams and of one individual from the genetic population of the genetic algorithm's team. I then run a large number of season simulations using the subroutine presented in Section 2.2 to calculate the expected metric for this individual. Then, I replace the individual used by another and reproduce the metric calculation until I have computed the metric of all the individuals from the genetic population. The fitness of an individual is the value of their metric.

[3] I actually like the idea of it not necessarily converging to the global optimum as it gives a "character" to the team and simulates the range of abilities of actual competitors

[4] I could have used only 3 genes, making use of the budget constraint to deduce the fourth budget investment but decided to code 4 genes to gain in flexibility

2.4.2 Selection

To discard unfit individuals, I rank them as a function of their fitness and discard the least fit individuals.

2.4.3 Breeding

The next stage in the genetic algorithm is to create as many individuals as have been discarded in the Selection stage[5]. To create an individual, thereafter called *offspring*, I randomly select two parents from the individuals who survived Selection. For each offspring gene, I randomly pick the corresponding gene from either of the parents.

Example: Breeding stage of the genetic algorithm

The stage of a genetic algorithm that I refer to as "Breeding" stage is conventionally taken care of by a crossover operation. The crossover operation normally consists in drawing an integer between 2 and the number of genes per individual and creating an offspring whose genes are those of the first parent when the gene number is strictly lower than the number drawn and those of the second parent for genes whose number is equal to or greater than the number drawn.

Rather than doing that, I am proceeding gene by gene, drawing, for each gene, from which parent the offspring gene comes from. Doing this (or any crossover operation for that matter) yields a temporary individual who violates the overall budget rule, so I rescale all the gene values to satisfy that rule. For example, the following situation might arise for an overall budget of 8.0¤:

	Gene			
Individual	Marketing	Reliability	Chassis	Engine
Parent 1	3.0	~~1.1~~	3.1	0.9
Parent 2	~~1.2~~	1.9	~~2.2~~	~~2.7~~
Temporary	3.0	1.9	3.1	0.9
Offspring	2.7	1.7	2.8	0.8

To avoid violating the overall budget constraint at this stage, I rescale each budget investment of the offspring. I proceed similarly for as many offspring as is necessary until the population is back to its original count.

2.4.4 Mutation

During the mutation stage, each gene has a probability of mutating, *i.e.*, of changing slightly. Such a mutation violates the overall budget constraint, so when I proceed to the mutation of one gene, I also select a second gene randomly and mutate it in a symmetric way to the first mutating gene. The mutation stage is applied to each individual: the offpsring but *also* the parents with the exception of the top-ranking individual. The counter-intuitive reason for which I included some of the parents to the mutation process is that they already contributed their genes during the Breeding stage (see Section 2.4.3) and were not the fittest individual of their generation. I believe that allowing them to mutate can substantially accelerate the evolution of the population towards fitter levels by avoiding to carry over useless information from one generation to another.

[5] this stage is normally called crossover but, as you will see, I am actually not doing crossover *per se*

2.4.5 Optional: selective individual reset

I also added an option to my genetic algorithm which is aimed at preventing it from getting stuck in local minima of little significance. This option takes place after the Mutation stage and simply consists in randomly selecting individuals who are not top-ranking and resetting their genes randomly.

2.4.6 Determination of the best response

The above stages are carried out in the order presented and, once a new generation is created (after the Mutation stage of Section 2.4.4 or the stage described in Section 2.4.5), the genetic algorithm increments the generation count and resumes computing from the Fitness calculation of Section 2.4.1.

The genetic algorithm loop constitutes only one stage of the wider iterative method presented in Section 2.3. As a result, it is, in practice, not necessary to make sure that the genetic algorithm converges to the optimal response and perfectly efficient to only run a small number of generations and not worry about the convergence of the computed response. Another way to understand this is that the convergence toward the optimal budget split is the task of the iterative method itself, so that the goal of this genetic algorithm is rather to make sure that the budget split of all the teams evolve in the right direction.

2.5 Artificial intelligence

I have, so far, assumed that all the competing teams play similarly: they optimize the same metric and are not limited in the budget they invest in any area. This is, of course, unrealistic and constitutes a major flaw in my algorithm.

To attempt to improve my decision-making algorithm, I decided to create an artificial intelligence (AI) inspired by budget splits taken from the competition history. Based on the way the historical budget splits were distributed, I created categories in Marketing and Chassis/Engine bias but not in Reliability, for which only one Season of history was available at the time I wrote my decision-making algorithm (and it seemed to indicate that the competitors chose their Reliability investment in a similar fashion). I compiled the data from the past season into budget split categories to determine the probability of a team having a certain AI profile. Then, I took the team lineup of the season for which I am trying to determine BURP's budget split and attributed to these teams (except BURP) an AI profile in such a way that the proportion of each profile reflects the historical data. An example of the way I attributed these AI profiles is shown below.

Season 4: Artificial Intelligence profiles

In order to design Artificial Intelligence profiles for Season 4, I first compiled the budget splits of all the teams participating in Seasons 1–3 and discarded two occurrences (one bot and one unusual budget split). By briefly looking at the investment patterns, I could identify certain investment categories. For Marketing, I could identify three such categories:

- low: lower than $0.6\mathfrak{i}$,
- intermediate: between $0.6\mathfrak{i}$ and $2.2\mathfrak{i}$,
- high: greater than $2.2\mathfrak{i}$.

I could also identify two categories in Chassis/Engine bias, which I computed using the following formula:

$$\frac{\max(\text{Chassis},\text{Engine})}{\text{Chassis} + \text{Engine}}.$$

The Chassis/Engine bias categories were:

- low: lower than 90%,
- high: greater than 90%.

I then collapsed the data from the past seasons into the following table:

Chassis/Engine bias	Marketing low	intermediate	high
low	2	3	11
high	5	4	3

As 14 teams are registered for Season 4, I created Artificial Intelligence profiles in proportion to the above table:

- Team A: BURP (unrestricted optimization)
- Team B: low/low
- Team C: low/intermediate
- Team D: low/high
- Team E: low/high
- Team F: low/high
- Team G: low/high
- Team H: low/high
- Team I: high/low
- Team J: high/low
- Team K: high/intermediate
- Team L: high/intermediate
- Team M: high/high
- Team N: high/high

The way I enforced these AI profiles is as follows. When I create new budget splits for AI teams, or when I modify these budget splits (as is necessary during certain steps of the genetic algorithm), I make sure that the constraints corresponding to the above categories are respected, so that all the budget splits corresponding to a team with a given AI profile fall in the relevant category as defined above.

Another way that I incorporated AI into my decision-making algorithm is through the use of different metrics. I first assumed all the teams optimize the metric I am interested in, which is, once again, not realistic. Then, I assumed that all the teams optimize the average number of points in the Constructor's Championship, except for BURP (which sticks to its own metric). Lastly, I assumed that a certain set of teams (which I varied too!) optimizes my metric while the other teams optimize the average number of points in the Constructor's Championship. In all the above AI setups, I calculated the optimal budget split using my decision-making algorithm. While these AI setups are not realistic on their own, they gave me a sense of the variability associated with the optimal budget split. I did observed different optima, but the differences between them are rather small and it was easy to come up with a choice that I used for BURP's budget split.

I would like to thank James Murkin: the idea of this Artificial Intelligence originates from discussions we had during his Master's project.

3 Program summary

Algorithm 1: BURP's budget split decision-making algorithm

```
   /* INITIALIZATION                                                    */
   /* Assume N teams for which an initial budget split and initial genetic
      algorithm population of budget splits need to be created          */
1  for Team = 1 to N do
2  |  Randomly create budget split for Team
3  |  Randomly create budget split for each individual of the genetic algorithm of Team
   |   // Note that the definition of ``randomly'' depends on AI
4  end
   /* ITERATIVE PROCESS                                                 */
5  while (Process is not converged) ∩ (Iteration number is too low) do
6  |  Simulate season
7  |  Store top-ranked budget split
8  |  for Team = 1 to N do
9  |  |  Set budget split of other teams to the one used during last season
10 |  |  Calculate best response of Team using genetic algorithm      // Note that various
   |  |   stages in the genetic algorithm need to be modified according to AI
11 |  end
12 |  for Team = 1 to N do
13 |  |  Replace budget split by best response
14 |  end
15 end
   /* POST-TREATMENT                                                    */
16 User looks at whether or not top-ranking budget split is converged
17 if (Convergence) then
18 |  Set BURP's budget split to top-ranking budget split
19 |  else
20 |  |  Generate distribution of top-ranking budget split
21 |  |  Set BURP's budget split to most frequent top-ranking budget split
22 |  end
23 end
```

4 The case of Season 4

I wrote the above code during the preparation for Season 3 and implemented the AI part (see Section 2.5) during the build up to Season 4, where I properly ran it for the first time.

The metric I used is the one explained in the lavender box of Section 2.1, the AI profiles are those identified in the lavender box of Section 2.5 and the rules of the competition (including the drivers available) are the ones used in the competition and available on the competition webpage.

To run the algorithm, a number of parameter values were chosen, mostly by common sense, but sometimes by proper testing. Here is the list of the parameter values used for the iterative method (see Section 2.3):

- Maximum number of iterations: 100
- Number of seasons simulated per call to the season simulation subroutine: $10,000$

Here is the list of parameter values used for the genetic algorithm (see Section 2.4):

- Population size for each team: 10
- Number of generations run per optimal response problem: between 5 and 20 to simulate various levels of intelligence
- Number of seasons simulated per call to the season simulation subroutine: $1,000$
- Selection ratio: 50%
- Mutation rate: 30%[6]
- Selective individual reset[7]: Yes
- Number of individuals reset: 2, drawn from the non top-ranking parent pool

I wrote my program in Fortran and compiled it using ifort with the option -fast, which proved to be the fastest of the optimization options I tested. One full execution of the program lasts at most 5h on my laptop (processor: Intel Core i5-8265). Although I ran my program mostly at night (the computer does not have to sleep) and when I am away from the keyboard, it is always entertaining and insightful to find out how the various teams evolve. Figure 1 shows the screen output that the program generates. This iteration shows that optimizing the metric does not provide the same result as optimizing the number of points in the Constructor's Championship. In fact, we learn from this iteration that low Marketing investments tend to maximize the average number of points in the Constructor's Championships but not the metric. The peculiarity of the metric is that it gives a heavy weight to the highest championship ranks which is far from being in proportion to the number of points scored. The lesson, here, is that investing little in Marketing leads to a high scoring team at the expense of chances to actually win the championship. Teams with a higher Marketing investment perform, on average, more poorly but tend to get a better shot at winning the championship. Another interesting observation from this iteration is that the unrestricted AI (BURP) does not always perform the best.

After running my iterative method a sufficient number of times, I decided to keep only the last 50 iterations for each run in order to discard misleading information generated by the random initial condition. I then calculated the frequency at which each possible investment appeared in the top-ranking budget split. Some of these results are reported in Figure 2. It is clear that the best Reliability investment is $1.4i$, which appeared more than 60% of the time in the top-ranking budget split. Similarly, the best Engine investment is $0.0i$, appearing more than twice as often as the second best option (46% against 19.8% for $0.1i$). The best choice for Marketing and Chassis is less clear, however, I decided to choose Marketing first, as it is the least ambiguous choice to make. I set the Marketing investment of BURP to $2.8i$, which appeared in 27.6% of the top-ranking budget splits I computed. BURP's Chassis investment

[6] Note that this is a probability drawn for each gene, so a lower bound for the actual mutation rate due to the way I applied the overall budget rule, see Section 2.4.4

[7] see Section 2.4.5

```
---------------------------
 Iteration           66
---------------------------

Rk      Team                Metric     AvgPts    Mkt Rel Cha Eng
 1      High/high 2         2.2787    102.8732   2.9 1.4 3.7 0.0
 2      High/high 1         2.2325    104.3709   3.0 1.4 3.6 0.0
 3      BURP                2.2303     99.7260   2.8 1.4 3.7 0.1
 4      Low/high 4          2.1433     95.9524   2.7 1.4 3.4 0.5
 5      Low/high 3          2.1082     99.5437   3.0 1.5 2.7 0.8
 6      Low/high 2          1.9788     92.8055   2.7 1.3 3.5 0.5
 7      Low/high 5          1.8900    100.3422   3.0 1.3 1.7 2.0
 8      Low/high 1          1.8662    100.2835   3.3 1.5 2.7 0.5
 9      High/low 2          1.4325    119.2133   0.1 1.2 0.0 6.7
10      Low/inter           1.3946     73.5106   2.2 1.6 2.4 1.8
11      High/low 1          1.3937    118.5170   0.1 1.2 0.0 6.7
12      High/inter 2        1.0682    112.4982   0.9 1.4 4.1 1.6
13      High/inter 1        0.9805    110.0171   0.8 1.5 5.3 0.4
14      Low/low             0.0025     70.3231   0.0 1.6 1.6 4.8
```

Figure 1: Snapshot of the screen output of my program during execution. The top line indicates the iteration number to keep track of progress. The table below shows the expected result of the last season simulated. The teams are identified by their AI profile (BURP being unrestricted). The Metric column refers to the metric I want to maximize and, thus, the ranking is made based on this column. The AvgPts column displays the expected number of points the team will score in the Constructor's Championship. The last four columns represent the budget split of the corresponding team, where Mkt (resp. Rel, Cha and Eng) corresponds to Marketing (resp. Reliability, Chassis and Engine). Note that, for this simulation, I had relaxed one AI condition: I deactivated the "Chassis/Engine bias high" condition application. This can be seen on the team called "High/inter 2", for which the Chassis/Engine bias is 72% instead of the required 90% minimum.

was then set to satisfy the overall budget rule: 3.8. This investment was actually the second most likely Chassis investment from my results: it was observed 18.0% of the time, against 19.6% of the time for its neighbor, 3.7. In Season 4, BURP therefore ran:

- Marketing: 2.8
- Reliability: 1.4
- Chassis: 3.8
- Engine: 0.0

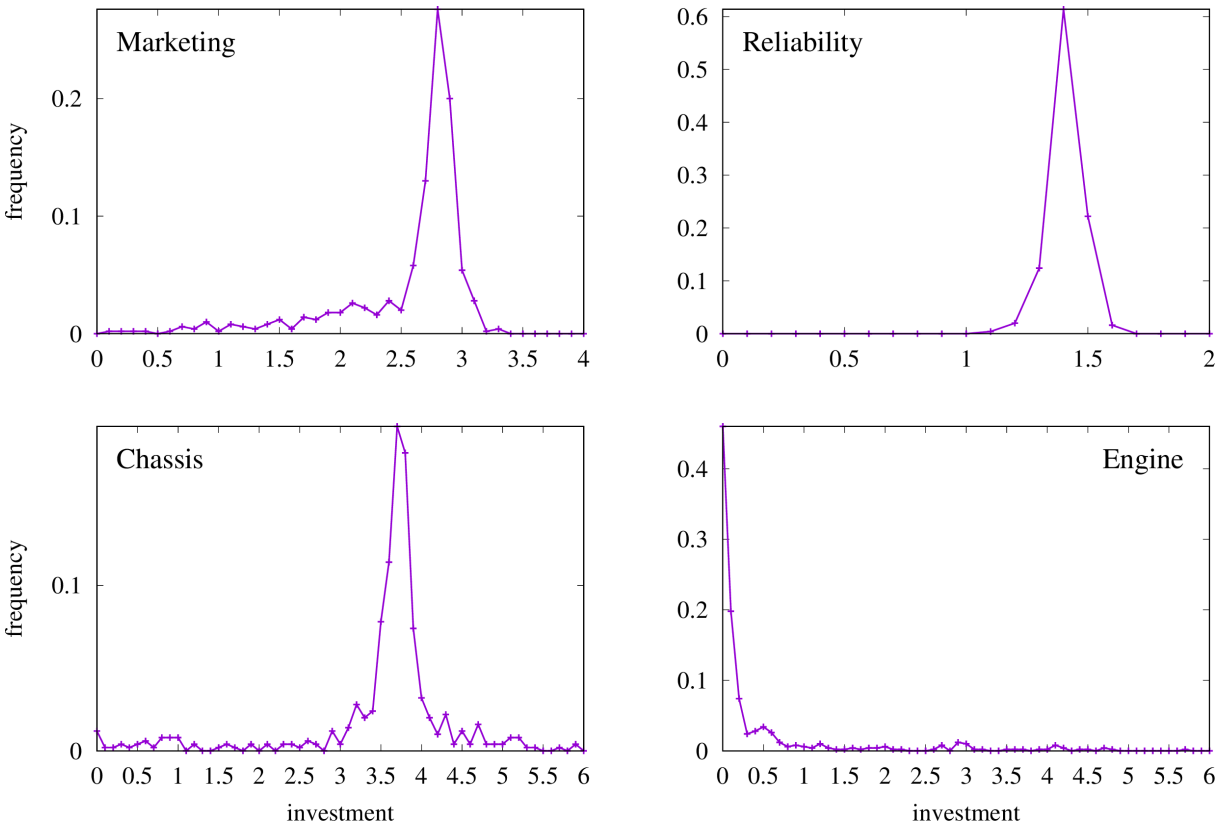

Figure 2: Investment frequency for each of the budget areas (Marketing, Reliability, Chassis and Engine) among the top-ranking budget splits taken during 16 runs of my iterative method (i.e. 800 data points—I only kept the last 50 iterations of each run) as a function of the investment value.