Racing strategy for the dynamic-customer location-allocation problem.

ANKRAH, R., LACROIX, B., MCCALL, J., HARDWICK, A., CONWAY, A. and OWUSU, G.

2020

Racing Strategy for the Dynamic-Customer Location-Allocation Problem

[1]School of Computing Science and Digital Media, Robert Gordon University, Aberdeen, Scotland
[2]Research and Innovation Department, British Telecommunications Plc, Ipswich, England

Reginald Ankrah[1] r.b.ankrah@rgu.ac.uk
Benjamin Lacroix[1] b.m.e.lacroix@rgu.ac.uk
John McCall[1] j.mccall@rgu.ac.uk
Andrew Hardwick[2] andrew.hardwick@bt.com
Anthony Conway[2] anthony.conway@bt.com
Gilbert Owusu[2] gilbert.owusu@bt.com

***Abstract*—In previous work, we proposed and studied a new dynamic formulation of the Location-allocation (LA) problem called the Dynamic-Customer Location-allocation (DC-LA) problem. DC-LA is based on the idea of changes in customer distribution over a defined period, and these changes have to be taken into account when establishing facilities to service changing customers distributions. This necessitated a dynamic stochastic evaluation function which came with a high computational cost due to a large number of simulations required in the evaluation process.**

In this paper, we investigate the use of racing, an approach used in model selection, to reduce the high computational cost by employing the minimum number of simulations for solution selection. Our adaptation of racing uses the Friedman test to compare solutions statistically. Racing allows simulations to be performed iteratively, ensuring that the minimum number of simulations is performed to detect a statistical difference.

We present experiments using Population-Based Incremental Learning (PBIL) to explore the savings achievable from using racing in this way. Our results show that racing achieves improved cost savings over the dynamic stochastic evaluation function. We also observed that on average, the computational cost of racing was about 4.5 times lower than the computational cost of the full dynamic stochastic evaluation.

***Index Terms*—Dynamic Customer Location-Allocation (DC-LA) Problem, Robust optimisation over time (ROOT), Dynamic stochastic evaluation function, Population-Based Incremental Learning Algorithm (PBIL), Simulation Model, Racing**

I. Introduction

The location-allocation problem involves choosing facility locations in a region of concern to service the demands of customers aimed at minimising total costs [1]. The fundamental characteristics of Location-Allocation (LA) problems require that any rational model reflect some aspects of future uncertainty [2]. Changes in the population growth and migration, market size, environmental factors and advancement in technology often drive the need of consumers which causes demand to vary over time. For this reason, facilities are expected to be effective in servicing demand over an extended planning period. Especially in cases where considerable capital and resource investment is required in establishing facilities, it is vital to take into consideration potential variations in demand over time when deciding the location of facilities [3]. To address this problem, we introduced a new dynamic formulation of Location-allocation problem in [4] that takes into account the actualised servicing costs and movement of customers over time. We called this problem the Dynamic-Customer Location-allocation (DC-LA) problem.

In this model, customers are located in cities, and their movement over time is driven by the randomly generated attractivity of each city.

DC-LA is formulated in the context of Robust optimisation over time (ROOT) [5] in that facilities are established once at the start of the planning period and are expected to be satisfactory in servicing changing customer locations.

In [4], we generated 1440 problem instances of the problem by varying three problem parameters: (1) movement rate, which determines the mobility of customers, (2) the number of facility locations (problem dimension) and (3) the number of customers. We compared the performance of Population-Based Incremental Learning Algorithm (PBIL) [6] using a static evaluation which assumes no customer movement, and a dynamic evaluation (referred to as a fixed number of simulations) which evaluates the fitness of a solution over 100 customer movement scenarios. Comparison of the results shows that the dynamic evaluation obtains better results but at a high computational cost.

To address this issue of high computational cost, in this paper, we apply the concept of racing [7] to reduce the number of simulations required for solution selection within our evolutionary framework. Racing uses a statistical test to compare solutions in the population within a race after they have been evaluated against several simulations. A race here is a single iteration of the search process. Racing is also dynamic in nature and evaluates solutions against a simulation just as the dynamic evaluation function does, however, instead of using a fixed number of simulations as is done in the dynamic evaluation function, racing initially evaluates solutions against a minimum number of simulations at the start of the race

This project is funded by British Telecommunications Plc. in collaboration with the Robert Gordon University

and performs a statistical test to discard the worst solutions from the population. If more simulations are required to eliminate weak solutions, only the required minimum number of simulations are generated by racing. Which helps to save on the number of evaluations. Racing was first proposed in machine learning to deal with the problem of model selection [7] and later adapted by [8] [9] for the tuning of optimisation algorithms. Racing has then been successfully applied to tackle real-world problems [10]–[13]

The rest of the paper is organised as follows. In section II, we give a summary of the DC-LA. In section III, we define the concept of racing. In section IV and V, we describe the experimental setup and discuss the result. Finally, we conclude and present future works in section VI.

II. Dynamic-Customer Location-Allocation Problem

Dynamic-Customer Location-Allocation (DC-LA) Problem involve the location of facilities to adequately service the changing distributions of customers over a defined period to reduce the overall total costs. In defining DC-LA, let a set of m cities $A = \{a_1, a_2, ..., a_m\}$ be a set of m potential locations, and $B = \{b_1, b_2, ...b_n\}$ be a set of n customers. Each $a_i \in [0, 1]^2$ and $b_j \in [0, 1]^2$ define the coordinates in a 2-dimensional plane. Here, the cost d_{ij} of connecting customer b_j to location a_i is defined by the euclidean distance between b_j to location a_i. Let a set $T = \{t_1, t_2, ..., t_{max}\}$ denote the defined period where t_{max} is the maximum length of time. DC-LA considers the potential movement of customers over a given period of T. In DC-LA, the pattern of customer location changes is assumed to be stochastically driven by the attractivity of cities. The attractivity of cities aims at modelling the shift in customer population. In DC-LA a facility is assumed to be located within the centre of a city. Hence for this paper, a city and facility will be used interchangeably to refer to the location of a facility. Locations employed in DC-LA are assumed to be discrete, and the optimal number of facilities are found by solving the problem. In this paper, the problem formulation assumes that facilities offer similar services to customers and facilities are unconstrained in the number of customers they can adequately service. The cost of opening a facility is set to the number of customers divided by the number of facilities.

DC-LA is formulated as Robust optimisation over time (ROOT) problem [5]. In robust optimisation over time, we aim to obtain solutions that are reliable or robust over the defined time, rather than pursuing the shifting optima. A solution is described as robust over a specific period when its quality continues to be satisfactory and is relatively indifferent to the environmental fluctuations during the defined time interval.

To generate an instance of this DC-LA, we first uniformly generate cities location and their attraction rate randomly. Based on those locations, customers are then iteratively generated by randomly selecting a city (based on the attraction rates). The coordinates of the customer are then obtained by sampling its location from a normal distribution centred in the coordinates of the city.

A. Measuring the goodness of a solution

The dynamic evaluation function considers the potential movement of customers over a given period T. This will influence the connection cost $C_t(x)$ in each t period. The decision variables are represented by a binary string $x = (x_1, ..., x_m) \in \{0, 1\}^m$ where 1 represents an opened facility and 0 represent a closed facility. The objective function of the dynamic evaluation function is thus formulated as:

$$f_{dynamic}(x) = \sum_{i=1}^{m} c_i x_i + C_0(x) + E\left[\sum_{t=1}^{t_{max}} C_t(x)(1+dr)^{-t}\right] \tag{1}$$

Where C_0 is the connection cost of each customer to an opened facility.

$$C_0(x) = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} x_{ij} \tag{2}$$

where $x_{ij} = 1$ if customer j is connected to facility i and $x_{ij} = 0$ if not.

Subject to:

$$\sum_{i\ =1}^{m}\sum_{j\ =1}^{n} x_{ij} = 1 \tag{3}$$

the cost function $C_0(x)$ is a deterministic function and returns the service costs of customers at time t_0. $E\left[\sum_{t=1}^{t_{max}} C_t(x)(1+dr)^{-t}\right]$ is the expected service costs $C_t(x)$ of customers for times $\{t_1, t_2, ..., t_{max}\}$ discounted over t_{max} using dr. dr is a *discount rate*, typically applied to allow comparison of costs incurred at different times. Because facilities are assumed to be able to service all the demand of a customer, Constraint 3 ensures that a customer is connected to only one facility.

B. Simulation Model

The simulation model is based on the assumption that customers will move over time. Each simulation starts by generating new attraction rates for each city. For each customer, we then generate the times at which the customer is going to move over the next t_{max} years. For this purpose, we introduce a new parameter call movement rate mr ranging from 0 to 1, which regulates the mobility of customers. The movement times of each customer are sampled from a normal distribution centred in $mr \cdot t_{max}$. Hence, the lower the movement rate, the higher the number of movements a customer makes.

Each simulation consists of generating customer movements and calculating their service costs over the period. The steps of a simulation are outlined in Algorithm 1.

Algorithm 1 Simulation Model

Require: A, B, t_{max}
  for Each Simulation **do**
    Generate attraction rate a' for each city $a_i \in A$
    for Each customer $b_j \in B$ **do**
      Set of movement dates: $M = \emptyset$
      $t = 0$
      while $t < t_{max}$ **do**
        $t = t + \mathcal{N}(mr \cdot t_{max}, 0.1 \cdot t_{max})$
        $M = M \cup t$
      end while
      for t_y in 1 to t_{max} **do**
        if $t_y \in M$ **then**
          Choose a new city for the customer based on A'
          Generate new location for customer in the new city
          Update cost for servicing customer based on open facilities
        end if
        Add cost of servicing customer b_j to total cost for year t_y
      end for
    end for
    Actualise costs obtained for t_{max} using discount rate dr
  end for

III. Racing as a model selection

The concept of racing was first developed in [7] by Maron et al. for model selection. Racing worked by testing the various models in parallel, one test point at a time. In this way, a running average could be maintained for each model's generalisation error. The average generalisation error is an estimate of the model's exact generalisation error had it been tested on all of the test points. By using a statistical bound, the closeness of the estimated generalisation error to the exact error could be determined. After a small number of test points, the best models, i.e. the models with the lowest generalisation error can be distinguished from the worst models (.i.e models with the highest generalisation error). The models that are significantly worse than the best ones are discarded from the race. The more test points that are observed, the tighter the estimated generalisation error is to the exact error. Hence many models can be differentiated from each other and discarded, thereby concentrating the computational effort on differentiating among the better model [7].

Racing was later adapted in [8] as Iterated racing to automatically configure optimisation algorithms. The process of iterated racing primarily involves three steps: (1) sampling new configurations according to a particular distribution, (2) selecting the best configurations from the newly sampled ones utilising racing, and (3) updating the sampling distribution in order to bias the sampling towards the best configuration. These three actions are iterated until a termination condition is reached [9].

From a general perspective, an iterated racing approach is any method that repeats the creation of solutions using a racing algorithm to select the fittest solutions. Therefore an exploration method of an iterated racing algorithm could be distinct from the current method of finding the best solutions and instead employ population-based algorithms or local searches. The essential factor here is the suitable combination of a search method with an evaluation that considers the underlying stochasticity of the evaluation into account [8]. Base on this reasoning, we are motivated to adopt the concept of iterated racing to the problem of Dynamic-Customer Location-Allocation (DC-LA) to help reduce the total number of evaluations and in general, the considerable computational effort expended by the dynamic evaluation function. In section III-A, we describe our adaptation of racing to DC-LAP.

A. Adaptation of Racing to Dynamic-Customer Location-Allocation problem

We employ the concept of racing [7] in DC-LA as a selection process to quickly discard the statistically worse solutions from the best solutions at the early stages of the search process thereby concentrating the computational effort on differentiating among the better solutions. An essential aspect of our adaptation of racing is in the truncation mechanism which strives to use the least number of simulations to compare solutions in the population.

In describing our adaptation of racing to DC-LAP, we first define the input parameters:

- A population size $|\mathcal{P}|$ of k solutions
- S_{max}: defines the maximum number of customer movement simulations per race. In the situation where solutions become mutually statistically indistinguishable, the race will continue to evaluate solutions against new simulations until the maximum number per race S_{max} is exhausted.
- S_{min}: minimum number of simulations before running a statistical test.
- A truncation rate μ: Based on the size k of the initial population, the race terminates when the size of the population $\mathcal{P}$ is decreased to μk. $\mu \in \{0, 1\}$

Once the initial population $\mathcal{P}$ of k solutions are generated by PBIL, for each race i every solution x in the population $\mathcal{P}$ is evaluated on a customer movement scenario S_i by $f(\mathcal{P}_x, S_i)$. Before a statistical test is performed, each solution has to have performed S_{min} simulations. We employ the Friedman test as the statistical test for determining statistical differences between the solutions. Friedman test is a non-parametric test that can be applied to sample of unknown distribution. Here, every solution in the population is tested on the same scenarios. Once a statistical difference has been recorded, the solution(s) considered to be statistically worst when compared to the best solution in the population are removed from the population. If neither of the terminating criteria for the race has been satisfied, i.e. S_{max} or μk, the race continues by generating a new scenario and evaluating the remaining solutions against the new scenario. After the

first statistical test has been performed using S_{min} simulations, subsequent statistical tests are performed more frequently after the single evaluation of all remaining solutions on every new scenario. After every test, statistically worse solutions from the best solutions are discarded from the population. Race continues until the size of $\mathcal{P}$ is decreased to μk or S_{max} is reached.

An example of a race is shown in figure 1. In figure 1 there exist 10 solutions. At every step of the race, the solutions are evaluated on a single scenario A' based on a new attraction rate. After several steps, those solutions that are deemed to be statistically worse than the best solution in the population are discarded from the population, and the race proceeds with the surviving solutions. Because the initial elimination test is essential in performing the statistical test, typically a higher number of simulations (A'_{first}) are observed before making the initial statistical test. Succeeding statistical tests are performed for each (A'_{each}) scenario. The process proceeds until a termination criterion is reached, i.e. a set minimum number of solutions in the population is reached, or the set maximum number of scenarios is exhausted, or a set number of solutions have been evaluated. Each node is the evaluation of a solution on a scenario. '_' indicates that no statistical test is performed, '+' indicates that the test removed at least one solution from the population, '=' indicates that the test did not remove any solutions from the population. In the example below, A'_{first} = 5 and A'_{each} = 1.

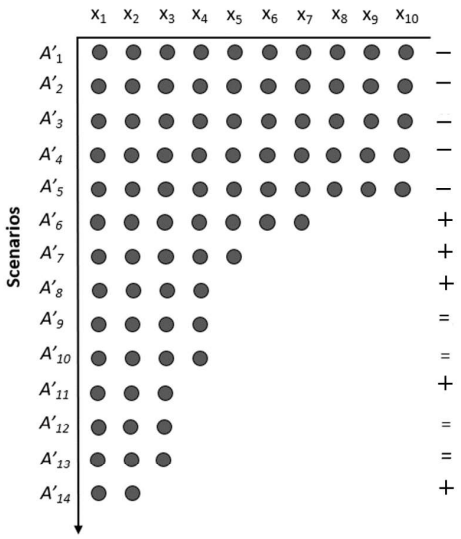


Fig. 1: Racing for solution selection.

After the race, PBIL updates the probability vector PV with the surviving solutions. PBIL then generates new solutions to reset the population to its initial value of k. Solutions surviving from the previous generation are carried on to the next generation. Because the surviving solutions are not re-evaluated on the same customer movement scenarios, it allows for the Algorithm to save simulations in further generations. Algorithm 2 shows the pseudo-code of racing.

Algorithm 2 Racing as a selection method

```
k : population size
μ : truncation rate
S_min : minimum number of simulations before running statistical test
S_max : maximum number of simulation per race
Generate initial population at random of k solutions P = {x_1, x_2, ..., x_k}
Generate set of customer movement scenarios S = {S_1, ..., S_max}
while termination criterion not reached do
  i=0;
  while |P|> μk AND i < S_max do
    i = i + 1
    for j in 1 to k do
      Evaluate F_ij = f(x_j, S_i)
    end for
    if i ≥ S_min then
      Perform statistical test on F_ij
      Remove from P all x_j that are significantly worse than the best individual in the population.
    end if
  end while
  Update probability vector of PBIL with remaining solutions.
  Generate new solution from PBIL probability vector.
  Add new solutions to P until the size of |P|= k
end while
```

IV. Experimental Setup and results

In this Section, we analyse the benefits of using the racing evaluations over the dynamic approach, which evaluates a solution by averaging the costs over a fixed number of simulations. Following our previous paper, the number of simulations was fixed to 100 simulations, and we will refer to this approach as f_{100}. It will be opposed to the racing approach which we will refer to as f_{racing}.

The parameters and problem instances used in our experiments are the same as in [4]. We generated 48 problem configurations by varying:

- The number of facilities $m = \{10, 20, 50, 100\}$.
- The number of customers $n = \{100, 500, 1000\}$.
- The movement rate $mr = \{0.25, 0.5, 0.75, 1\}$.

For each of the 48 configurations, we generated 30 instances, giving us a total of 1440 problem instances. Each run is allowed to evaluate 10000 solutions. At the end of each run, the best solution is evaluated using the fixed number of simulation evaluation over 5000 simulations, and the expected cost of that solution is returned. For each problem, the average over 20 runs is retained for comparison. The following sections present and discuss the results obtained.

A. DC-LAP parameters influence on results

In order to better understand the problem characteristics, we analyse the performance of each approach according to the parameter values used in each configuration.

We present a summary of wins and ties of racing (f_{racing}) against the fixed number of simulation evaluation (f_{100}) in table I.

From table I, we observe that the maximum number of wins is obtained by f_{racing} on problems of larger dimensions (higher number of facilities).

A look at the problems configurations with a smaller number of facilities shows more ties recorded between f_{racing} and a fixed number of simulation evaluation. However, f_{racing} is seen to achieve more wins on average than the fixed number of simulation evaluation. It is also observed that for a smaller number of facilities, i.e. $m10$ especially in situations where customers make frequent movements over the planning period, f_{racing} might not be the better choice of the two evaluations as it achieves fewer wins than f_{100}.

To see how the wins achieved by f_{racing} translates into cost savings, we direct our attention to figures 2a, 2b and 2c which shows the percentage cost savings between f_{racing} and f_{100} grouped according to the movement rate mr, the number of facilities m and number of customers n respectively. Negative values mean cost savings.

In figure 2a, f_{racing} is seen to improve the costs achieved by the fixed number of simulation evaluation of up to about 0.5% across all customer movement scenarios. The performance of f_{racing} on movement rates shows f_{racing} to be the better choice among the two evaluation functions when consideration is made to the movement of customers in deciding the locations of facilities.

In figure 2b, f_{racing} is observed to improve on the cost savings achieved by the f_{100} for a smaller number of facilities of about 0.125% and this increases to about 0.55% for a larger number of facilities.

In figure 2c, f_{racing} is observed to improve on the savings achieved by f_{100} with an improved cost savings of about 0.55% for a smaller number of customers and decreases to about 0.35% for a larger number of customers. The improved cost savings of f_{racing} makes it the better evaluation function when compared to f_{100} concerning the number of customers.

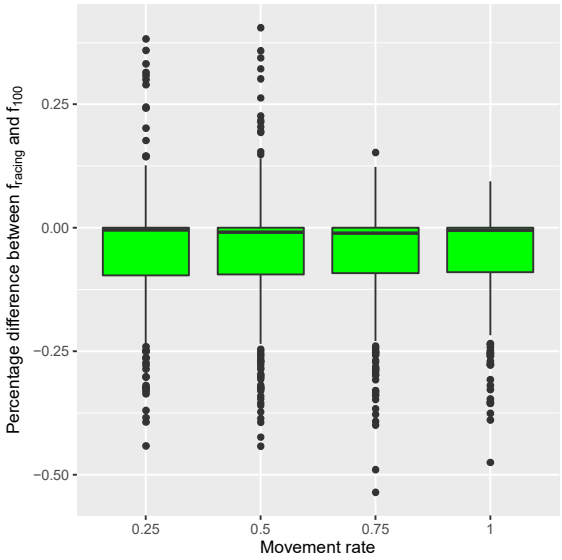
Results discussed in this Section shows that f_{racing} achieves improved cost savings when compared to the fixed number of simulation evaluation concerning all problem parameters of movement rate mr, the number of facilities m and number of customers n. An essential reason of adapting f_{racing} to our problem was to help reduce the number of simulations needed to efficiently compare solutions in a population during the search process thereby reducing the considerable computational effort that comes with evaluating solutions with many simulations as done in f_{100}. We, therefore, examine the computational effort expended by f_{racing} in terms of time on the DC-LA instances in section IV-B.

B. Computational time complexity

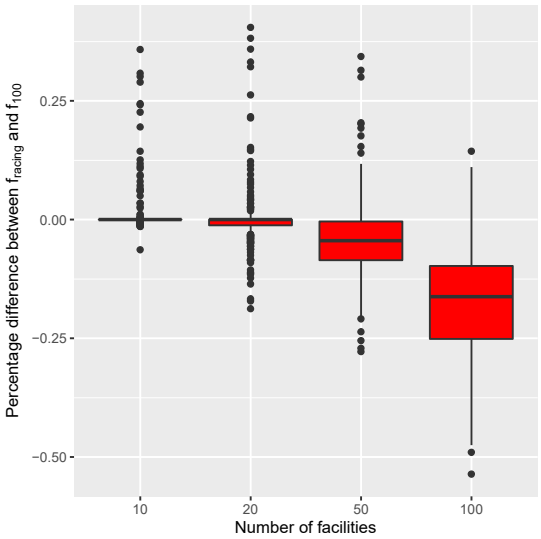
In this Section, we examine the computational time of f_{racing} on problem instances based on the problem parameters of DC-LAP.

TABLE I: Wins, Losses and Ties of f_{racing} and f_{100} grouped by configuration of DC-LA

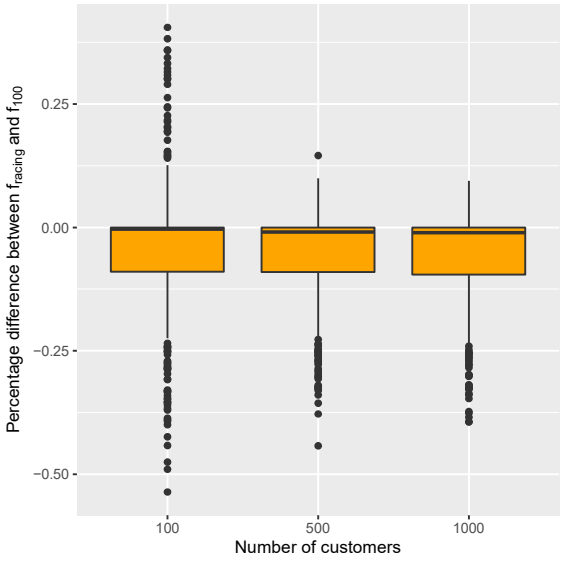
mr	m	n	f_{racing}	f_{100}	Ties
0.25	10	100	3	8	**19**
0.5	10	100	3	9	**18**
0.75	10	100	5	4	**21**
1	10	100	0	1	**29**
0.25	10	500	7	2	**21**
0.5	10	500	6	2	**22**
0.75	10	500	2	1	**27**
1	10	500	1	1	**28**
0.25	10	1000	2	1	**27**
0.5	10	1000	5	0	**25**
0.75	10	1000	2	3	**25**
1	10	1000	1	2	**27**
0.25	20	100	**14**	12	4
0.5	20	100	**14**	10	6
0.75	20	100	**12**	6	**12**
1	20	100	9	0	**21**
0.25	20	500	**13**	7	10
0.5	20	500	**15**	6	9
0.75	20	500	13	1	**16**
1	20	500	9	0	**21**
0.25	20	1000	**16**	3	11
0.5	20	1000	**17**	4	9
0.75	20	1000	12	1	**17**
1	20	1000	9	0	**21**
0.25	50	100	12	**17**	1
0.5	50	100	**15**	14	1
0.75	50	100	**26**	3	1
1	50	100	**25**	5	0
0.25	50	500	**22**	8	0
0.5	50	500	**24**	6	0
0.75	50	500	**23**	7	0
1	50	500	**26**	4	0
0.25	50	1000	**26**	4	0
0.5	50	1000	**24**	6	0
0.75	50	1000	**28**	2	0
1	50	1000	**27**	3	0
0.25	100	100	**22**	8	0
0.5	100	100	**28**	2	0
0.75	100	100	**27**	2	1
1	100	100	**27**	3	0
0.25	100	500	**28**	2	0
0.5	100	500	**28**	2	0
0.75	100	500	**30**	0	0
1	100	500	**29**	0	1
0.25	100	1000	**30**	0	0
0.5	100	1000	**29**	1	0
0.75	100	1000	**29**	1	0
1	100	1000	**28**	2	0



(a) Grouped by movement rate



(b) Grouped by number of facilities



(c) Grouped by number of customer

Fig. 2: Percentage difference between the f_{racing} and f_{100} grouped according to DC-LA parameters

Results from experiments show that as customers make frequent movement over time, the average computational time of f_{racing} is 4.8 times lower than the average time recorded by the fixed number of simulation evaluation. Also, as customers make little or no movement over time, the average time recorded by f_{racing} is 4.5 times lower than the average time recorded by f_{100}.

For the number of facilities, we observe that on the smallest number of facilities, the average time recorded by f_{racing} is 4.5 times lower than the average time recorded by f_{100} and the average time recorded by f_{racing} on the larger number of facilities is 4.7 times lower than the average time recorded by f_{100}.

For the number of customers, we observe that on the smallest number of customers, the average time recorded by f_{racing} is 4.3 times lower than the average time recorded by the f_{100} and the average time recorded by f_{racing} on the larger number of customers is 4.8 times lower than the average time recorded by f_{100}.

The improved computational time of f_{racing} can be attributed to the ability of f_{racing} to discard weak solutions at the beginning of the search process through the use of statistical tests to compare solutions in the evolutionary framework. This approach allows simulations to be performed iteratively until a statistical difference is reached, which ensures that the minimum number of simulations is performed to detect statistical difference to support solution selection. On the other hand, due to the larger number of simulations required by the fixed number of simulation evaluation, a considerable effort is often wasted in the early stages of the search process on weak solutions. The waste of effort and a large number of simulations all contribute to the expensive computational cost of f_{100}, which is on average 4.5 times higher than the average time recorded by f_{racing}.

Overall problem parameters the ratio between the variance in computational time recorded for f_{racing} is 24 times lower than the variance in computational time recorded by f_{100}. The variance shows that the computational time recorded for f_{racing} on problem configurations are relatively closer to the mean recorded time than for f_{100}. On average, f_{racing} employs about 21 simulations within each race of a run to quickly discard weak solutions from the population. The use of the minimum amount of simulation to discard weak solutions accounts for the low variance in the computational costs recorded for f_{racing}.

V. The Maximum Likelihood Solution

So far, in this paper, the solution returned by the search was the one that yields the best-expected fitness over a limited number of simulations. In this Section, we are interested in comparing that solution with what we call the *Maximum Likelihood Solutions* (MLS), *i.e* the solution that is the most likely to be generated. As an EDA, PBIL builds a probabilistic model from the most promising solutions and uses it to sample new candidate solutions.

The probability of obtaining MLS is defined as:

$$P(x) = \prod_{i=1}^{m} p_i(x_i) \tag{4}$$

Because the probability vector (PV) in PBIL is a univariate model, i.e. it assumes full independence of problem variables the probability of MLS: *P(MLS)* is the product of the probabilities of the individual properties. Although the probability of obtaining MLS is minimal, MLS has a whole region around it that is quite similar to it. Taking the region into consideration gives us a large concentration of probabilities. We employ MLS for comparison to see if we can get an estimate consistent with results obtained by an evaluation function when we run the experiments many times and hence avoid running many experiments. To allow for comparison of results, we evaluate MLS with the same 5000 scenarios used to evaluate the best solutions found for f_{racing}.

TABLE II: Recorded wins and ties between f_{racing} and $f_{racingMLS}$ on problem configurations

mr	m	n	f_{racing}	$f_{racingMLS}$	Ties
0.25	10	100	**19**	3	8
0.5	10	100	**17**	3	10
0.75	10	100	**19**	0	11
1	10	100	**23**	0	7
0.25	10	500	**24**	0	6
0.5	10	500	**21**	0	9
0.75	10	500	**24**	0	6
1	10	500	**27**	0	3
0.25	10	1000	**21**	0	9
0.5	10	1000	**24**	0	6
0.75	10	1000	**23**	1	6
1	10	1000	**25**	0	5
0.25	20	100	**21**	4	5
0.5	20	100	**18**	8	4
0.75	20	100	**21**	1	8
1	20	100	**21**	0	9
0.25	20	500	**21**	1	8
0.5	20	500	**22**	1	7
0.75	20	500	**21**	0	9
1	20	500	**24**	0	6
0.25	20	1000	**21**	2	7
0.5	20	1000	**23**	1	6
0.75	20	1000	**23**	1	6
1	20	1000	**22**	0	8
0.25	50	100	**18**	11	1
0.5	50	100	**18**	11	1
0.75	50	100	**30**	0	0
1	50	100	**30**	0	0
0.25	50	500	**29**	1	0
0.5	50	500	**30**	0	0
0.75	50	500	**30**	0	0
1	50	500	**30**	0	0
0.25	50	1000	**30**	0	0
0.5	50	1000	**30**	0	0
0.75	50	1000	**30**	0	0
1	50	1000	**30**	0	0
0.25	100	100	0	**30**	0
0.5	100	100	0	**30**	0
0.75	100	100	0	**30**	0
1	100	100	1	**29**	0
0.25	100	500	3	**27**	0
0.5	100	500	2	**28**	0
0.75	100	500	3	**27**	0
1	100	500	2	**28**	0
0.25	100	1000	3	**27**	0
0.5	100	1000	5	**25**	0
0.75	100	1000	2	**27**	1
1	100	1000	3	**27**	0

The results observed in table II shows that for larger problems having 100 facilities, using the MLS offers better results than the results obtained by f_{racing}. This means that for much larger problems, we can employ the MLS as a measure of making decisions to locate facilities without having to run many experiments.

VI. Conclusion

In previous work [4], we introduced a new dynamic variant of Location-allocation problem called the Dynamic-customer Location-allocation (DC-LA) problem where facilities are opened once at the start of a defined period and are expected to be satisfactory in servicing customers demands irrespective of changes in customer distribution. To help evaluate a solution to DC-LA, we explored a stochastic dynamic evaluation function in the context of simulation-based optimisation, which we called f_{100} or a fixed number of simulation evaluation. f_{100} takes into account possible movements of customers over the planning period. It does this by simulating customer movements to estimate the expected cost over time. To evaluate the performance of f_{100}, we employed a static evaluation function that forms the baseline for comparison. The static evaluation function, which is a deterministic function assumes that customers will make no movements over the planning period and thus, their locations remain the same from start to the end of the planning period.

Although observations from experiments showed that using a fixed number of simulation evaluation led to better costs savings when compared to the static evaluation function, the fixed number of simulation evaluation came with a high computational cost due to a large number of simulations required in the evaluation process.

To help achieve a balance between a large number of simulation and the high computational cost, we adapted the concept of f_{racing} as a selection method to our problem. f_{racing} was first proposed in machine learning to deal with the problem of model selection and later adapted for the configuration of an optimisation algorithm. In recent years, f_{racing} has been adapted in solving optimisation problems.

Our adaptation of f_{racing} uses the Friedman test to compare solutions in PBIL statistically. f_{racing} allows simulations to be performed iteratively, ensuring that the minimum number of simulations is performed to detect a statistical difference.

The experiments conducted with f_{racing} on the same problem instances employed in our previous work showed f_{racing} to have better results when compared to the fixed number of simulation evaluation on all DC-LA parameters. In terms of cost savings, f_{racing} showed good performance by achieving improved cost savings over f_{100}.

We also observed that on average, the computational cost of f_{racing} was about 4.5 times lower than the computational cost recorded for f_{100}. Results obtained shows that not only is f_{racing} able to reduce a large number of simulations required to select solutions within an evolutionary framework efficiently, but f_{racing} also achieves better results and improved cost savings due to the ability of f_{racing} to quickly discard weak solutions from the population at the early stage of the search process. The application of f_{racing} can be adapted

to other aspects of location problems with a stochastic cost function.

A study of the maximum likelihood solution (*MLS*) showed that for problem configurations with a larger number of facilities (i.e. 100 facilities) we could employ the MLS to decide the locations of facilities without having to run many experiments; thereby saving much computational effort.

In this work, we employed the Friedman test for testing for the statistical difference between solutions in f_{racing}. The statistical power of the Friedman test is dependent on the sample size, i.e. a larger sample size has more statistical power. Hence, before we can perform the initial statistical test in the race, we have to evaluate each solution in the population several times to give us enough of a sample size for testing. Future work will explore other non-parametric statistical tests other than the Friedman test to see if we can find statistical differences between solutions with a lesser number of simulations.

Also in this work, we used PBIL algorithm to solve DC-LA problem however in the literature other EDA's that are bi-variate in nature has been shown to improve on the performance of uni-variate EDA's such as PBIL especially in considering the strong trend in customer movements of DC-LA problem. This is because univariate EDAs treat each decision variable independently and hence, they are often not representative enough to provide the best performance. Future work will, therefore, explore bi-variate EDA such as the h-BOA [14] to see if allowing some dependencies between variables in DC-LA problem can help to further reduce the number of evaluations. Finally, we will seek to compare the performance of racing with other approaches from the literature.

REFERENCES

[1] Scott AJ. Location-allocation systems: a review. Geographical Analysis. 1970;2(2):95–119.

[2] Farahani RZ, Abedian M, Sharahi S. Dynamic facility location problem. In: Facility Location. Springer; 2009. p. 347–372.

[3] Farahani RZ, Asgari N, Heidari N, Hosseininia M, Goh M. Covering problems in facility location: A review. Computers & Industrial Engineering. 2012;62(1):368–407.

[4] Ankrah R, Lacroix B, McCall J, Hardwick A, Conway A. Introducing the dynamic customer location-allocation problem. 2019;.

[5] Fu H, Sendhoff B, Tang K, Yao X. Robust optimization over time: Problem difficulties and benchmark problems. IEEE Transactions on Evolutionary Computation. 2015;19(5):731–745.

[6] Ankrah R, Lacroix B, McCall J, Hardwick A, Conway A. A Holistic Metric Approach to Solving the Dynamic Location-Allocation Problem. In: International Conference on Innovative Techniques and Applications of Artificial Intelligence. Springer; 2018. p. 433–439.

[7] Maron O, Moore AW. The racing algorithm: Model selection for lazy learners. Artificial Intelligence Review. 1997;11(1-5):193–225.

[8] Birattari M, Stützle T, Paquete L, Varrentrapp K. A racing algorithm for configuring metaheuristics. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. Morgan Kaufmann Publishers Inc.; 2002. p. 11–18.

[9] López-Ibáñez M, Dubois-Lacoste J, Cáceres LP, Birattari M, Stützle T. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives. 2016;3:43–58.

[10] Lacroix B, McCall J, Lonchampt J. Iterated racing algorithm for simulation-optimisation of maintenance planning. In: 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2018. p. 1–7.

[11] Becker S, Gottlieb J, Stützle T. Applications of racing algorithms: An industrial perspective. In: International Conference on Artificial Evolution (Evolution Artificielle). Springer; 2005. p. 271–283.

[12] Hoos HH. Automated algorithm configuration and parameter tuning. In: Autonomous search. Springer; 2011. p. 37–71.

[13] Birattari M, Yuan Z, Balaprakash P, Stützle T. F-Race and iterated F-Race: An overview. In: Experimental methods for the analysis of optimization algorithms. Springer; 2010. p. 311–336.

[14] Pelikan M. Hierarchical Bayesian optimization algorithm. In: Hierarchical Bayesian Optimization Algorithm. Springer; 2005. p. 105–129.