Iterated racing algorithm for simulation-optimisation of maintenance planning

Benjamin Lacroix
School of Computing and Digital Media
Robert Gordon University
Aberdeen, Scotland
b.m.e.lacroix@rgu.ac.uk

John McCall
School of Computing and digital Media
Robert Gordon University
Aberdeen, Scotland
j.mccall@rgu.ac.uk

Jérôme Lonchampt
Industrial Risk Management Department
EDF Lab
Chatou, France
jerome.lonchampt@edf.fr

***Abstract*—The purpose of this paper is two fold. First, we present a set of benchmark problems for maintenance optimisation called VMELight. This model allows the user to define the number of components in the system to maintain and a number of customisable parameters such as the failure distribution of the components, the spare part stock level and every costs associated with the preventive and corrective maintenances, unavailability and spare parts. From this model, we create a benchmark of 175 optimisation problems across different dimensions. This benchmark allows us to test the idea of using an iterated racing algorithm called IRACE based on the Friedman statistical test, to reduce the number of simulations needed to compare solutions in the population. We assess different population size and truncation rate to show that those parameters can have a strong influence on the performance of the algorithm.**

***Index Terms*—Maintenance optimisation, racing, statistical test**

I. Introduction

Maintenance planning is an optimisation problem which consists in maximising the Net Present Value (NPV) of the maintenance strategy of an asset or multi-component systems. A maintenance strategy can go from a simple scheduling of maintenance operations [1], maintenance intervals [2] or a more holistic approach taking into account spare part stock management and logistics [3].

It is a highly trending topic in the field of Engineering Asset Management (EAM) and industries of all kind recognise it as a major challenge for reducing their operation costs. In the recent review [4], the authors list recent research in the domain of maintenance optimisation. It however appears that most research in this is domain is done by the EAM community and little interest is given to it by the computational intelligence community. As a consequence, these research focus on the modelling side of maintenance problems more than on the ways to solve them. Search algorithms commonly used are based on classic schemes such as Genetic Algorithms [5] or Simulated Annealing [6].

The cost function of these problems is based on models able to evaluate the integrity of a proposed strategy. These models simulate the life cycle of a system going through a series of discrete events such as failures, maintenances and spare parts acquisition. These models are usually driven by a stochastic events such as the component failures.

Most metaheuristics are developed to optimise deterministic cost functions where the comparison of solutions is straightforward. Little work exist on the optimisation stochastic problems.

In this context, two solutions have been identified. The first one called the direct method consists in using a statistic of Monte Carlo simulations (mean, median) [7]–[9]. The problem with this approach is the uncertainty of its computational cost. Indeed, it requires a large number of simulations which can sometimes be very computationally expensive and requires an insight on number of simulations needed. This leads to a large effort being wasted in the early stage of the search process on poor solutions. The second approach, the indirect method, uses surrogate functions to approximate the stochastic code [10], [11]. Here again, the reliability of the meta model is questionable. Generally speaking, the main issue when dealing with stochastic code in a cost function is to know if the comparison of two solutions or a population of solution is reliable and cost effective.

To tackle this problem, we propose to use statistical tests to compare solutions in an evolutionary framework. The advantage of such test in the context of simulation optimisation is that they can be performed iteratively. Indeed, the random sample issued from the simulation can be augmented iteratively until statistical difference is reached. This ensures that the minimum number of simulations is performed to detect statistical difference to support solution selection.

Based on this idea, we propose in this paper to use iterated racing as the selection method. Racing was initially proposed in [12] for model selection in machine learning and was later developed in a successful search algorithm for off-line parameter tuning in IRACE [13].

In order to test the proposed method, we also present a new generic benchmark developed in collaboration with *Élecricité de France* (EDF) and based on a simplified version of the simulation tool VME presented in [14] called VMELight. We generate 175 instances of the VMELight problem by randomly varying the parameters of the failure probability distribution of the components at hand.

This paper is organised as follows. In Section II, we present

This project is funded the Fondation Mathematique Jacques Hadamard through the PGMO program

the VMELight maintenance optimisation and the benchmark proposed for the experiments of this paper. In Section III, we explain the IRACE algorithm used for maintenance planning. In Section IV we provide the results obtained by IRACE on the benchmark problems along with an analysis of the behaviour of the algorithm. Finally, we conclude with a list of open questions and future works.

II. Maintenance Optimisation

In this section, we describe the problem used for the experiment presented in this paper. We first describe the life cycle model used to evaluate maintenance plans and how we generate instances of this problem to create the benchmark.

A. The maintenance problem

For this paper, we use the simplified version of the simulation tool called VME presented in [14]. We refer to this problem as VMELight. The objective is to find the maintenance dates $P^* = \{p_1^*, p_2^*, ..., p_n^*\} \in \mathbb{R}^n$ that maximise the Net Present Value (NPV) of the maintenance strategy of n components. The failure probability of each asset follows a Weibull distribution.

The simulations are driven by the event model shown in Figure 1. The decision variables P are defined in the green event box and the probabilistic event of failures in the red event box. When a maintenance (replacement) event occurs, the model checks for the availability of a spare part. If not available, then a spare part is supplied. Planned and corrective maintenances assume a complete rejuvenating of the component by replacing it with a spare part in stock. If the stock is empty a spare part is acquired which causes a delay in the maintenance. Each event has a cost and a time frame as explained in the following section.

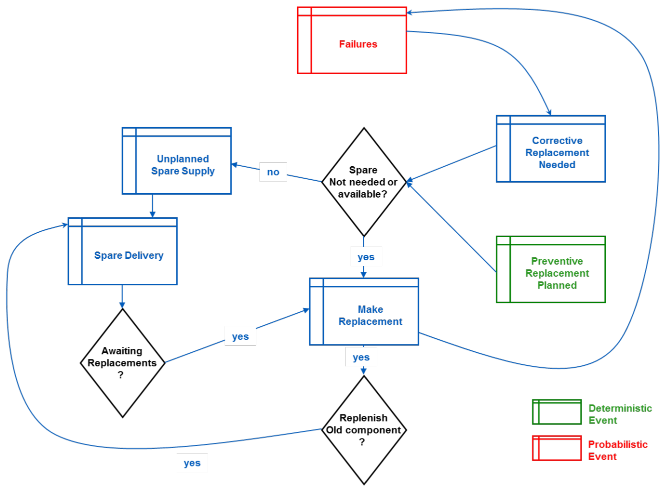


Fig. 1: Event model of VMELight

The NPV is calculated by comparing the proposed plan P against a reference strategy which consist in never performing any planned maintenance. For each simulation, the failure times of each component are generated independently from the given failure probability distribution of each components. We call this a failure scenario $T = \{t_1, ..., t_n\}$ where $t_i = W(\lambda_i, \beta_i, \gamma_i)$ where W is a Weibull distribution which defines the reliability of a component. The Weibull distribution is a common choice for this kind of problem.

Figure 2 illustrates the comparison between the reference strategy and an assessed strategy $P = \{p_1, p_2\}$ given a a failure scenario $T = \{t_1, t_2\}$. Since we consider that the a preventive maintenance reset the state of the component to as good as new, if a preventive maintenance p_i occurs before a failure on component i (*i.e.* $p_i < t_i$), the new failure time t_i^* is shifted by $t_i^* = t_i + p_i$. In this example, the planned maintenance on the second component shifts the failure beyond the life span of the asset, preventing it to occur.

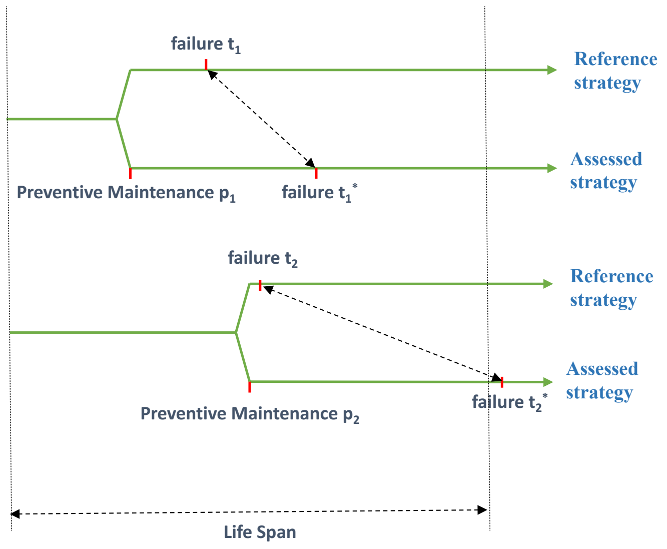


Fig. 2: Simulation example for two components

In the example provided by EDF on which this paper is based, the n components are equally used in two separate systems ($n/2$ components per system), see Figure 3. As a consequence, if the failure of a component will cause a own time of the whole system. If another component fails while its system is down, the failure will have no additional effect on the NPV. In this problem every component share the same spare part stock.

B. The benchmark

An instance of the VMELight problem is defined by the following parameters:

- Number of components n
- The failure probability follows Weibull distribution for which the parameters are defined in the following ranges:
  - $\Lambda = \{\lambda_1, ..., \lambda_n\} \in [0.01, 0.1]^n$
  - $B = \{\beta_1, ..., \beta_n\} \in [1, 4]^n$
  - $\Gamma = \{\gamma_1, ..., \gamma_n\}$ always equal to 0
- Components age at $t = 0$ is set to zero, which means that when we start the simulation, each component is considered as new.

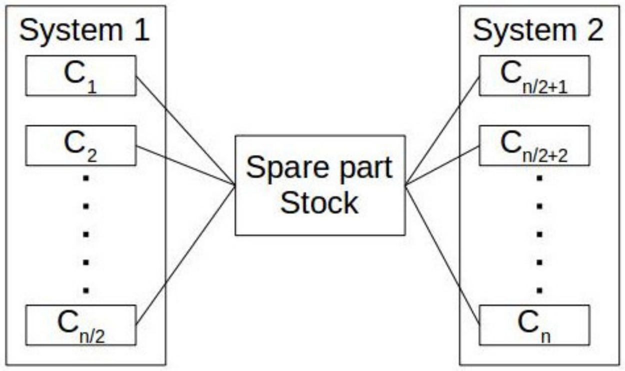


Fig. 3: Dependencies between components

- Time horizon is 40 years. Hence, the search space is defined in $\Omega \in [0, 40]^n \in \mathbb{R}^n$.
- Stock level is the number of spare parts available at the beginning of the simulation.It is set to $0.25n$
- Yearly cost of unavailability: $C_{unavailable} = 10000$
- Failure cost $C_{failure} = 1000$
- Maintenance cost $C_{maintenance} = 100$
- Spare part cost $C_{spare} = 10$
- Storage cost per year: $C_{storage} = 1$
- Time to buy a spare part after failure $t_{failure} = 1$
- Preventive maintenance time $t_{maintenance} = 1$
- Actualisation rate is 0.075.

Each of this parameter can be modified at will to reflect a specific case. In practice, failure probability distribution parameters are either provided by the manufacturer or can be fitted on past failure events. For this paper, we use the values indicated above and varied only the failure distribution parameters of the components by uniformly generating random values for the λ and β parameters. The decision of keeping the same values for costs is driven by the wish to ease the comparison between the different problems and different dimensions. We have generated 25 instances for seven different numbers of components $n = \{4, 8, 12, 16, 20, 30, 40\}$, corresponding to the dimension of the problem. This provides us with a benchmark of 175 instances.

III. Iterated racing for maintenance planning optimisation

In this section, we present our version of the iterative racing algorithm called IRACE. Racing is a comparison method used when the cost function requires multiple simulations. In the case of machine learning, as it was originally proposed, racing is used to select the best models for a training set [12]. In algorithm configuration [13], racing is used to compare sets of parameters for an optimisation algorithm against a set of problems or instances of a problem. In this case, we use racing to compare maintenance plans against a set of failure scenarios obtained from the different characteristics of a system and its components to maintain.

A. *Iterated racing*

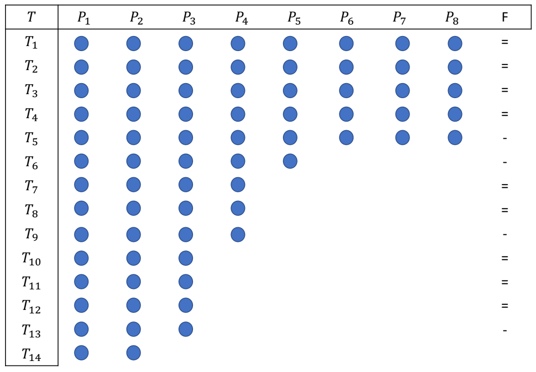
IRACE is a generational EA which originality lies in the truncation mechanism which aims at using the minimum number of replications in simulation optimisation where the cost function is stochastic or a statistic of a set of replications. Search algorithms and particularly metaheuristics are all based on the evaluation and comparison to support decisions on whether discarding them or using them to generate new solutions.

IRACE starts by uniformly sampling an initial population $\mathcal{P}$ of k solutions from the decision space Ω and a set of M failure scenarios $\mathcal{T}$ from the failure distribution of the n components defining the problem.

The truncation phase is called a race. At each step i of the race, each remaining solutions j in the population is evaluated on a given failure T_i by $f(P_j, T_i)$. It consists in iteratively evaluating each solution on the failure instances previously generated until statistical difference is obtained. When a solution is considered significantly worse than the rest of the population, it is removed from the population and the race carries on with the remaining ones. We consider two stopping criterion for a race:

- The truncation rate μ: given an initial population size k, the race stops when the population size $|\mathcal{P}|$ is reduced to μk
- The maximum number of failure scenario M: In order to prevent races to stall if for instance the population has converged, it is best to set a maximum number of replications per solution. Once this number is reached, we can assume that the number of replications done is sufficient to use the mean $\overline{NPV}$ as a reliable indicator to discard the remaining extra solutions from the population.

An example of a race is illustrated in Figure 4. In this example, a race between eight solutions takes place. After the simulation of the fifth failure scenario (T_5), the three solutions P_6,P_7 and P_8 are considered significantly worse and are discarded. The race carries only simulating the remaining solutions until only two solutions survive ($\mu k = 0.25 \times 8$).



T	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	F
T_1	●	●	●	●	●	●	●	●	=
T_2	●	●	●	●	●	●	●	●	=
T_3	●	●	●	●	●	●	●	●	=
T_4	●	●	●	●	●	●	●	●	=
T_5	●	●	●	●	●	●	●	●	-
T_6	●	●	●	●	●				-
T_7	●	●	●	●					=
T_8	●	●	●	●					=
T_9	●	●	●	●					-
T_{10}	●	●	●						=
T_{11}	●	●	●						=
T_{12}	●	●	●						=
T_{13}	●	●	●						-
T_{14}	●	●							

Fig. 4: Example of a race for a population size of $k = 8$ and a truncation rate of $\mu = 0.25$

At the end of each race, new solutions are generated from the surviving ones to reset the population to its initial value k. Solutions surviving from the previous generation are carried on to the next one. It allows the algorithm to save simulations in further generations. In this work, we use a simple Univariate Marginal Distribution Algorithm (UMDA) which samples each parameter of new solutions from a normal distribution obtained from the mean and standard deviation of the remaining solutions in the population $\mathcal{N}(\overline{p_i}, \sigma(p_i))$.

Algorithm 1 EA based on *racing* selection

```
Require: $k$ : population size
    $\mu$ : truncation rate
    $m$ : minimum number of simulations before running statistical test
    $M$ : maximum number of simulation per race
    $\Omega$ : the search space
1: Generate initial population at random of k maintenance plans $\mathcal{P} = \{P_1, ...P_k\}$
2: Generate set of failure scenarios $\mathcal{T} = \{T_1, ...T_M\}$
3: while termination criterion not reached do
4:     while $|\mathcal{P}| > \mu k$ AND $i < M$ do
5:         $i = i + 1$
6:         for j in 1 to k do
7:             Evaluate $NPV_{ij} = f(P_j, T_i)$
8:         end for
9:         if $i \geq m$ then
10:            Perform statistical test on $NPV$
11:            Remove from $\mathcal{P}$ all individuals $p_j$ if significantly worse than the "best" individual in $\mathcal{P}$
12:        end if
13:    end while
14:    while $|\mathcal{P}| > \mu k$ AND $i < M$ do
15:        Remove solution with the lowest $\overline{NPV}$
16:    end while
17:    Generate new population from $\mathcal{P}$
18: end while
```

This model only uses two parameters, k, the population size and μ the truncation rate. For the rest of this paper, we use the notation IRACE($k+\mu$) to denote the iterated racing algorithm used in the experiments

B. Statistical test

In this paper, as it is recommended in [13], we use the Friedman's non-parametric two-way analysis of variance by ranks [15] as the statistical test. The Friedman test is based on the rankings of each solution on each failure scenario. Although the objective is to maximise the mean NPV, and though a statistical test based on this indicator (such as the Student *t-test*) could be more appropriate, it has not been proven that the NPV distribution of the cost function follows a normal distribution. It would make the test unreliable and thus the use of a non-parametric test is essential.

IV. Experiments

In this section, we present the experiments performed and the results obtained, providing an analysis on the behaviour of IRACE with respect to the different parameters applied.

A. Experimental setup

As explained in Section II, experiments in this paper are based on a benchmark of 175 instances of the VMELight problem. We test IRACE($k + \mu$), using the parameters listed in Table I. The purpose of these experiments is to analyse the effect of the population size and the truncation rate. We allow 500,000 simulations per run and we replicate each run 25 times. Each run has a different set of failure scenarios which remains consistent between the different parameter settings. The performance comparison is only done at the end of the run. When the 500,000 simulations are used, every solution in the population is evaluated by 10,000 simulations and the solution returned is the one with the highest mean NPV. Using the mean NPV to compare appears to be a common practice in EAM.

TABLE I: IRACE($k + \mu$) parameters

Parameter	Description	Value
k	Popoulation size	$\{20, 50, 100\}$
μ	Truncation rate	$\{0.25, 0.5, 0.75\}$
m	Minimum number of iterations per race	20
M	Maximum number of iterations per race	5000

B. Results

In this section We present the results obtained by the different versions of IRACE($k + \mu$). We first analyse the behaviour of the search by analysing the effect of the different parameters on the racing and the evolution of the population's diversity along the search. We then study the effect on the final results obtained by each configuration.

1) Diversity: Figure 5 shows the evolution of the diversity of the population along a run. The diversity measure used is the average pairwise Euclidean distance between solutions in the population. the evolution of the diversity is plotted against the number of simulations performed in the search. We only show here the convergence of the population for problems in dimension 40 to avoid redundant information as every other dimensions gives similar results. the diversity is averaged over every problem and every run for $n = 40$. First, we note that the population size has the most influence on the diversity. Smaller population sizes lead to a faster convergence of the population. Then, as we could expect, larger truncation rates maintain a higher diversity in the population while small truncation rate impose a strong selective pressure on the population causing a faster convergence.

2) Racing effect: To analyse the effect of racing on the behaviour of the algorithm, we record the number of generations each search goes through (Figure 6a) and the number of new solutions evaluated (Figure 6b) for the different configurations. Given the stopping criteria of a race (the population is reduced with respect to the truncation rate), races may use different number of simulations to complete. This affects the number of races done during a run and hence, the number of solutions explored.

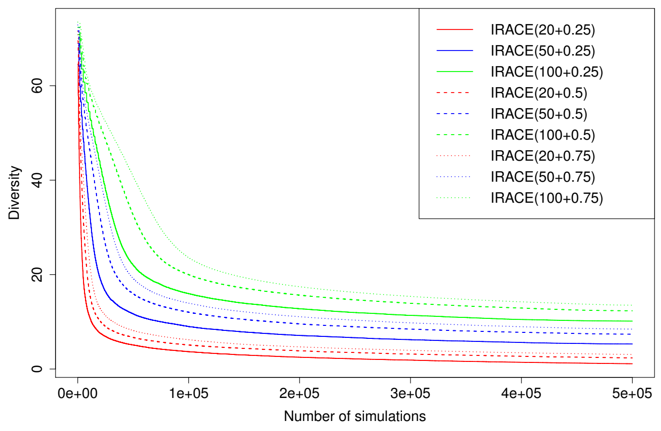


Fig. 5: Average diversity in the population through the runs for $n = 40$

The analysis of the race length and its effect on the number of solutions explored can bring an insight on the behaviour of the algorithm and the characteristics of the problem.

The truncation rate has a strong influence on the number of simulations needed to obtain statistical differences between the μ best solutions and the $1-\mu$ worse solutions. Indeed, the larger the truncation rate, the lesser the best solutions have to compete with each other. As a consequence, less simulations are needed to discard the $1-\mu$ worse solutions as it can be seen in Figure 6a. Where for every dimension, truncation rates of $\mu = 0.75$ (dotted lines) produces more races (generations) than $\mu = 0.5$ (dashed lines) and $\mu = 0.25$ (solid lines).

It is also interesting to note that the number of generations increases with the dimension despite the fact that the same number of simulations is allowed in each run. It can be explained by the fact that larger problems boast more disparity in their fitness landscape leading to an easier discrimination between solutions.

3) NPV: Based on the observation, we analyse the effect of the parameters on the final results obtained by each configuration. In Figure 7a, we show the mean NPV obtained by each version of IRACE on all 15 instances of each dimension and over each of the 25 runs. Figure 7b shows the mean ranking obtained by each configuration for each dimension.

First of all, we can see that the mean NPV obtained increases linearly with the number of components. The population size has the most influence in results as the three best configurations boast a population size of $k = 100$ (green lines in Figure 7 followed by $k = 50$ (blue lines) and finally $k = 20$ (red lines).

For the rest for this analysis, and in order to simplify the display of results, we discard the configurations with population size $k = 25$ and $k = 50$, to focus on $k = 100$.

Following the procedure described in [16], we use Friedman test and Holm's procedure to compare the results obtained by the three truncation rates $\mu = (0.25, 0.5, 0.75)$ applied to a population size of $k = 100$. In Table II, we show the mean rankings of the three configurations and apply the Friedman test. We obtain statistical differences for $\alpha = 0.05$ in dimensions $n = \{8, 12, 16, 20, 30, 40\}$. This allows us to apply the Holm's procedure to detect statistical differences between the best ranked configurations and the two others for these dimensions.

First, we note that there is no statistical difference between $\mu = 0.25$ and $\mu = 0.5$ in the lower dimensions ($n = \{8, 12, 16, 20\}$) while IRACE(100+0.75) obtains results that are significantly worse. This reflects the fact that the VMELight problem in low dimensions favours IRACE with a stronger selection pressure and faster convergence. On the other hand, when increasing the dimension, this tendency is reversed and configurations with less selective pressure obtain significantly better results. Indeed, for $n = 30$, there is no statistical difference between IRACE(100+0.5) and IRACE(100+0.75), while IRACE(100+0.25) is significantly worse than the former. Finally, in the largest dimension tested ($n = 40$), IRACE with a truncation size of $\mu = 0.75$ obtains significantly better results than both lower values of μ. Hence, when increasing the dimension, VMELight requires a broader exploration of the search space.

V. Conclusion

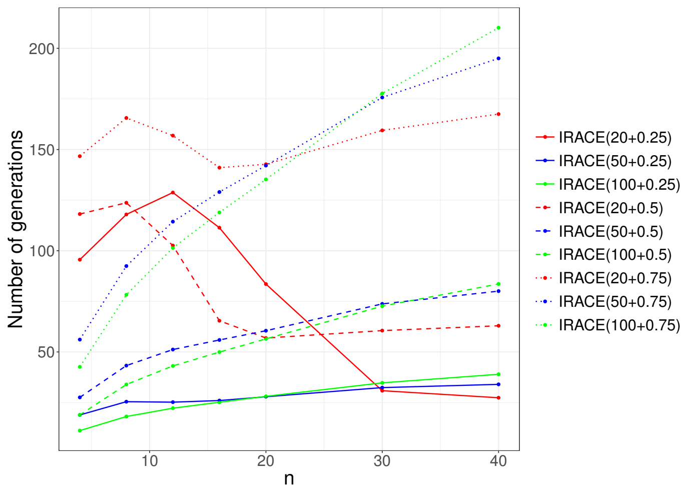
In this paper, we introduce the idea of using iterated racing to tackle maintenance optimisation problems. The idea of using IRACE for this kind of problems aims at removing the number of replications used in Monte-Carlo simulations to evaluate a maintenance strategy on a stochastic evaluation model.

To test the method, we proposed a new set of generic optimisation problem called VMELight. We generated 175 instances of this problem in seven different dimensions. We used this benchmark to test and analysed the behaviour of IRACE using different values for population size and truncation rate parameters. We showed that larger population size obtained the best results regardless of the problem and its dimension. We then observed that smaller problems preferred faster converging configurations while larger dimensions preferred explorative configurations.

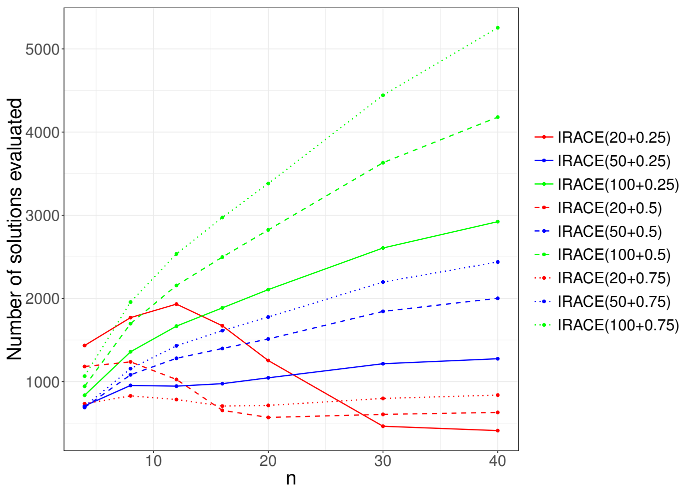
VI. Future works

Since maintenance problems have mainly been tackled by research from the EAM community, their focus was mainly on the simulation models and little on the search algorithm and methodology used to optimise these problems. Such problems, however present many interesting characteristics which should attract the attention of the computational intelligence community.

First, it would be interesting to develop a more flexible benchmark that will take into account more complex dependencies between the different components and different spare part stocks. At the moment, VMELight only considers one configuration (two systems and one shared spare part stock). This would allow the users to design more realistic and complex models. The benchmark used here is a first

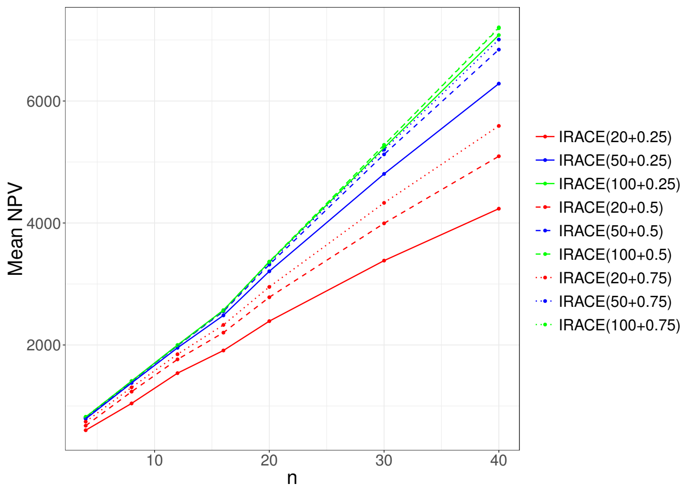


(a) Average number of generations per run

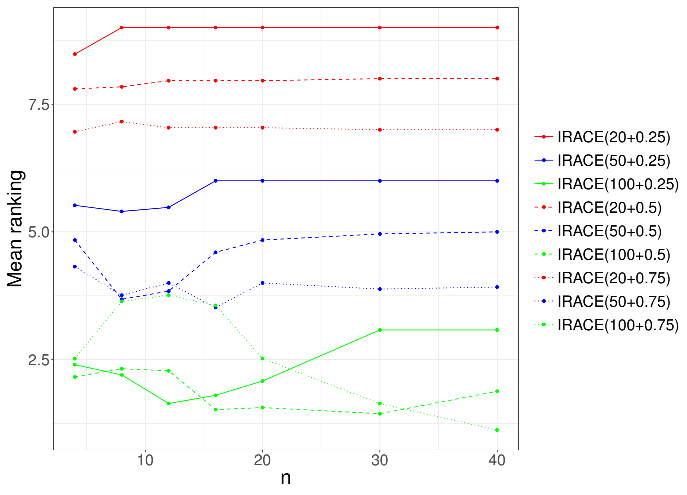


(b) Average number of solutions evaluated per run

Fig. 6: IRACE behaviour analysis for each configuration



(a) Mean NPV per dimensions



(b) Mean ranks per dimensions

Fig. 7: Mean NPV and mean ranking for each configurations in each dimension

TABLE II: Mean ranking and Friedman $p - value$

n	IRACE(100+0.25)	IRACE(100+0.5)	IRACE(100+0.75)	$p - value$
4	2.16	1.84	2	5.27E-01
8	1.8	**1.76**	2.44	**2.63E-02**
12	**1.6**	1.76	2.64	**3.94E-04**
16	1.68	**1.44**	2.88	**3.45E-07**
20	2.04	**1.56**	2.4	**1.18E-02**
30	2.92	**1.44**	1.64	**9.98E-08**
40	3	1.88	**1.12**	**2.29E-10**

TABLE III: Holm's procedure with $\alpha = 0.05$

n	Reference	i	Configuration	$z = (R_0 - R_i)/SE$	$p - value$	α/i	Statistical difference?
8	IRACE(100+0.5)	2	IRACE(100+0.75)	2.40	1.62E-02	0.025	Yes
		1	IRACE(100+0.25)	0.14	8.88E-01	0.05	No
12	IRACE(100+0.25)	2	IRACE(100+0.75)	3.68	2.36E-04	0.025	Yes
		1	IRACE(100+0.5)	0.57	5.72E-01	0.05	No
16	IRACE(100+0.5)	2	IRACE(100+0.75)	5.09	3.56E-07	0.025	Yes
		1	IRACE(100+0.25)	0.85	3.96E-01	0.05	No
20	IRACE(100+0.5)	2	IRACE(100+0.75)	2.97	2.98E-03	0.025	Yes
		1	IRACE(100+0.25)	1.70	8.97E-02	0.05	No
30	IRACE(100+0.5)	2	IRACE(100+0.25)	5.23	1.67E-07	0.025	Yes
		1	IRACE(100+0.75)	0.71	4.80E-01	0.05	No
40	IRACE(100+0.75)	2	IRACE(100+0.25)	6.65	3.00E-11	0.025	Yes
		1	IRACE(100+0.5)	2.69	7.21E-03	0.05	Yes

step towards creating an available framework to allow researchers from the computational intelligence community to study and develop new algorithms to tackle this kind of problems. Further steps thus include the standardisation of the evaluation method (number of simulation allowed, dimension, comparison method) to provide a fair research environment.

Consequently, further development on the pool of algorithms available for these problems is essential. More advanced and fine tuned heuristics are still to be proposed to improve the performance and knowledge on stochastic optimisation problems. For instance, in the context of the proposed algorithm and instead of the simple univariate model used here to sample new solutions, it would be interesting to implement different methods such as CMA-ES or Genetic Algorithms. Further research can also be done on the use of statistical tests in existing algorithms. Steady-State Evolutionary Algorithm such as Differential Evolution or direct search could for example use the Wilcoxon test for the pairwise comparison in their replacement. Other paradigm such as swarm intelligence could also be applied here.

Then, a whole new methodology still needs to be developed to understand the behaviour of algorithms tackling these problems, including convergence analysis, parameter sensitivity, scalability or run length distribution.

Finally, a need for parallelisation is of high interest for these problems since more complex life cycle models may require large computational effort. For that matter, IRACE is particularly adapted since simulations can be run independently.

References

[1] R. L. Cheu, Y. Wang, and T. F. Fwa, "Genetic Algorithm-Simulation Methodology for Pavement Maintenance Scheduling," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 6, pp. 446–455, nov 2004.

[2] D. Edwin, U. Gomez, S. Hennequin, and N. Rezg, "Optimization of a failure prone manufacturing system with regular preventive maintenance: an ipa approach," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10 422 – 10 427, 2011, 18th IFAC World Congress.

[3] W. Y. Yun, Y. J. Han, and G. Park, "Optimal preventive maintenance interval and spare parts number in a rolling stock system," in *2012 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, June 2012, pp. 380–384.

[4] A. Alrabghi and A. Tiwari, "State of the art in simulation-based optimisation for maintenance systems," *Computers & Industrial Engineering*, vol. 82, pp. 167–182, apr 2015.

[5] P. Lynch, K. Adendorff, V. Yadavalli, and O. Adetunji, "Optimal spares and preventive maintenance frequencies for constrained industrial systems," *Computers & Industrial Engineering*, vol. 65, no. 3, pp. 378–387, jul 2013.

[6] A. Alrabghi, A. Tiwari, and A. Alabdulkarim, "Simulation based optimization of joint maintenance and inventory for multi-components manufacturing systems," in *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, ser. WSC '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 1109–1119.

[7] M. Marseguerra, E. Zio, and L. Podofillini, "Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation," *Reliability Engineering & System Safety*, vol. 77, no. 2, pp. 151–165, aug 2002.

[8] P. Chootinan, A. Chen, M. R. Horrocks, and D. Bolling, "A multi-year pavement maintenance program using a stochastic simulation-based genetic algorithm approach," *Transportation Research Part A: Policy and Practice*, vol. 40, no. 9, pp. 725 – 743, 2006.

[9] J. Lonchampt and W. Lair, "Risk-informed simulation optimization for engineering asset management," 01 2014.

[10] A. I. J. Forrester, A. Sbester, and A. J. Keane, *Engineering Design via Surrogate Modelling*. Chichester, UK: John Wiley & Sons, Ltd, jul 2008.

[11] T. Browne, B. Iooss, L. L. Gratiet, J. Lonchampt, and E. Remy, "Stochastic simulators based optimization by gaussian process metamodels application to maintenance investments planning issues," *Quality and Reliability Engineering International*, vol. 32, no. 6, pp. 2067–2080, 2016, qre.2028.

[12] O. Maron and A. W. Moore, "The Racing Algorithm: Model Selection for Lazy Learners," *Artificial Intelligence Review*, vol. 11, no. 1/5, pp. 193–225, 1997.

[13] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[14] J. Lonchampt and K. Fessart, "Investments Portfolio Optimal Planning for industrial assets management: Method and Tool," in *ANS embedded meeting on Risk Management for Complex Socio-technical Systems (RM4CSS)*, Washington DC, USA, 2013.

[15] M. Friedman, "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[16] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, mar 2011.