Multiobjective Policy search for Robot Damage Recovery

Omar Samir Mohammed
Masters Student M1
MLDM Program
omar.mohammed@etu.univ-st-etienne.fr

Supervised by:
Prof. Jean-Baptiste Mouret
Permanent Researcher
Larsen team
INRIA Nancy
jean-baptiste.mouret@inria.fr

Contents

1 Introduction

Autonomous Robots are used in many fields, ranging from factories, critical rescue missions, space explorations, etc, just to count few. In the case of such complex systems, the robots may need to handle unexpected conditions while pursuing their mission. Legged robots clearly illustrate this need to handle the unexpected: to be as versatile as possible, they involve many moving parts, many actuators and many sensors. But, aside from these advantages, they can be damaged in many different ways. As such, a growing interest in the research community is to develop methods and algorithms for the robots to be able to recover from the damages on their own.

Many approaches have been developed in order to deal with the problem of robot damage and fault tolerance (Koos et al., 2013). For example:

- The traditional approach, which depends on identifying the faulty system, and having redundant components to deal with different failures (Visinsky et al., 1994; Koren and Krishna, 2007). This approach is complex and expensive, yet it had proven its quality in many fields, like space systems.
- Using robust controllers that can tolerate hardware and sensor inefficiencies (Goldberg and Chen, 2001; Lin and Chen, 2007).
- Finally, preparing some pre-designed solutions for the robot to cope with potential failures.

A new approach to deal with this issue consists in letting the robot learn on its own new and high-performing solutions. Several experiments have been done in this area (Connell, 1993). Many of this work was to tune the controllers of complex robots, and some of this work was to adapt the robot to new and unexpected situations (Koos et al., 2013). This internship project takes this approach.

Most of the work in this direction consisted in enabling the robot to find solutions that maximize one objective. For example, to enable to a hexapod robot to find a solution that will make it move as fast as possible. For the robot to learn on its own such solution, it is a long procedure (researchers report learning time from 90 min to several hours), and it is expensive procedure as well. A recent breakthrough in this area (Cully et al., 2015) enabled the robot

to find a satisfying solution in a very short time (2 min) with few trials on the robot itself.

However, in a damaged robot scenario, robots need to find solutions that balance between different multiple objectives. This is essential for the robot in order to perform its mission. For example, the robot may need to balance between speed and its body stability, in order to continue moving while, for example, being able to take photos, or performing the tasks required. A main critical issue is the speed of the recovery process itself. Multiobjective optimization usually takes considerable time and many evaluations on the robot in order to find a solution. Trials on the robot are very expensive and too slow to perform.

In this project, the target is to explore methods to extend existing work in both single objective optimization (Cully et al., 2015) and multiobjective optimization (Tesch et al., 2013) for robot damage recovery, in order to reach a diverse set of possible solutions that compromise between different objectives, taking into out account the need to find solutions as fast as possible, and with as few number of of trials on the real robot as possible.

To approach such problem, I start investigating different multiobjective optimization algorithms, performing tests on current benchmark problems, in order to ensure that we have the right framework, and get some sense with what these algorithms do. I then move on to test these algorithms on a simulated hexapod robot, with two targets in mind: to verify the results obtained from the benchmark (and that we can carry on with this), and to identify a possible efficient optimization algorithm that will be suitable for such expensive optimization.

This report will be divided into the following sections:

- **Current state of the art.** In this section, I will first introduce basic concepts like *Pareto Optimality* and *Hypervolume indicator*. Then, I explore the current advances in the field of robot damage recovery. A pioneer work in the area of single objective robot damage recovery will be discussed, and how they managed to reduce the speed of the recovery using *a prior* made in simulation. Then, I discuss the another research in the area of multiobjective optimization in robotics with an innovative multiobjective optimization approach based on the concept of *expected*

hypervolume improvement.

- **Experiments and Results.** In this section, I describe the experiment I have performed so far in my project. I start first by describing the results from testing the algorithms on benchmark problems, and drawing conclusions from this experiment. I then move on to the second experiment, where I test the different algorithms on a simulated robot, and I state the obtained results.

- **Future work.** Finally, I present the future steps to be done in my internship in order to achieve the stated goal.

2 Current state of the art

We start on this section by exploring the latest research in the area of single objective optimization for robot damage recovery (Cully et al., 2015). We then move on to the current work in multiobjective optimization (Tesch et al., 2013).

2.1 Single Objective Optimization for robot damage recovery

A number of experiments had been done on single objective robot damage recovery before. One of the noticed consequences for such process is the considerable trials needed to be done on the robot in order for the optimization algorithm to reach a satisfying solutions.

Researchers in the past gave a lot of attention to the issue of robot damage recovery. Many attempts have been made in order to achieve this process as soon as possible. Such task was usually been achieved in the previous researchers in more than an hour, and sometimes several hours.

In their work, researchers in (Cully et al., 2015) used a combination of *Bayesian Optimization* and a *prior* made in simulation, the researchers were able to minimize the time for a robot to recover from a damage from several hours to as low as 2 minutes. They produced a new algorithm, *Intelligent Trial and Error Algorithm*, which can be in pseudo-code as in Algorithm. 1

Algorithm 1 Intelligent Trial and Error Algorithm

```
1: procedure Intelligent Trial and Error Algorithm
2: Before the mission:
3:     Create Behavioral Repertoire (Via the MAP-ELITES algorithm in simulation)
4:     while In mission do
5:         if Significant performance fall then
6:             Damage Recovery Step (Via RBOA Algorithm)
```

One of the important concepts descried in this paper is **Bayesian Optimization** (Brochu et al., 2010). Bayesian Optimization techniques are considered to be some of the most efficient optimization methods, from point of the number of evaluations needed. Its efficiency mainly is a result from

using prior belief about a problem to chose the next point to sample.Bayesian Optimization is a black-box optimization. It starts with a smooth response surface. The goal is to fit this surface to our objective function - in this case, the objective function is the robot speed -, FIGURE 1. We can also see that the variance is flat in this case.

Figure 1: A flat response surface, without fitting anything to it yet (Source of this image is `http://yelp.github.io/MOE/demo_tutorial.html`).

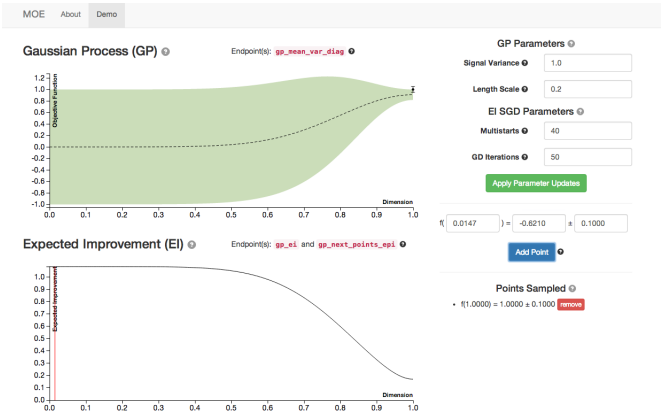

Figure 2: The response surface after fitting one point, and the variance changes associated with it. (Source of this image is `http://yelp.github.io/MOE/demo_tutorial.html`).

When fitting that surface with a single data point, the variance in this point is decreased to noise only (the response surface belief about this point is stronger), and also the variance around this point decreases as well, figure 2. You can also see the expected objective function has changed to adapt for the new data point. With multiple data points being fit to the response surface, it become more representative to the objective function. We get FIGURE 3

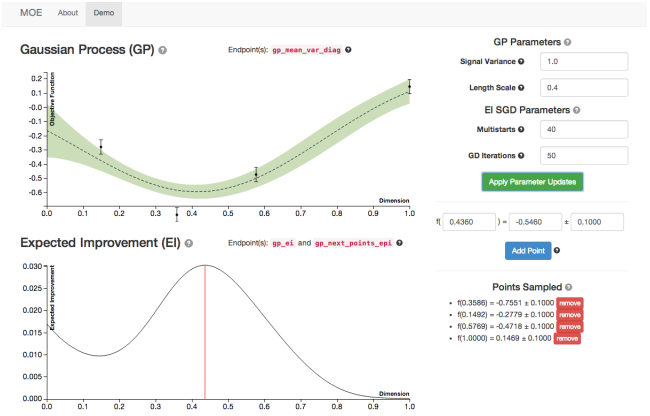

Figure 3: The response surface after fitting more points(Source of this image is `http://yelp.github.io/MOE/demo_tutorial.html`).

2.2 Expensive multiobjective optimization for robotics

The second work that inspired this project is the work done in (Tesch et al., 2013). In this work, the researchers discuss a novel approach for multiobjective optimization for applications where each evaluation is expensive, and it's important to reach a good Pareto front with the least number of evaluations.

Then, they apply this approach to robotics. They experiment with a snack robot, where they perform multiobjective optimization for two objectives: maximize speed of the gait of the robot and also maximize the stability of the snack robot head.

The approach the researchers follow in the paper is that they fit a surrogate function (a Gaussian Process (Rasmussen, 2006) in this case), one for each objective, with sampling points. With enough sampling points (each point will be tried on the robot and its performance will be recorded), they get a good

model for the objective functions. The authors then use these models to select the next promising point to try on the robot, record its performance, and then they update their models, and repeat the process, until they reach a satisfying solution, FIGURE 4. There approach in more detail:

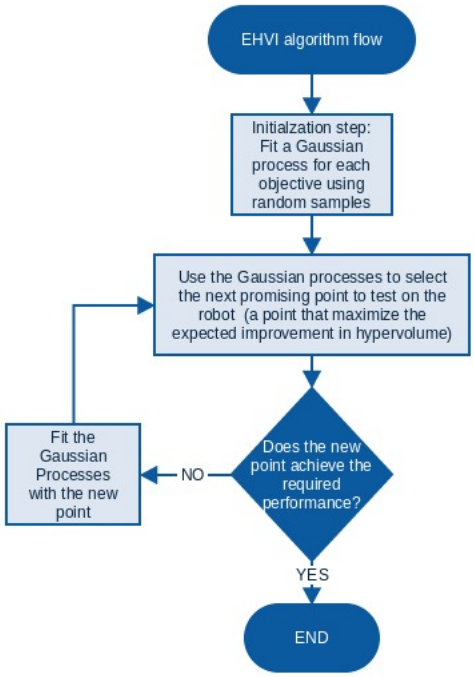

Figure 4: Flowchart summarising the approach of expensive multiobjective optimization, used in (Tesch et al., 2013)

- **Pareto Optimality.** In multiobjective optimization, we represent the values of the objective functions as vector $f : R^n \rightarrow R^m$. m is the objective vector dimension (number of objectives), and n is the decision vector x dimension (the dimension of the input vector), and $m > 1$. In case of minimization problems, we want to minimize

$$y = f(x) = (f_1(x), ..., f_m(x))$$

where $x \in R^n$, and $y \in R^m$. A solution y_a dominates the solution y_b if $y_{ai} \leq y_{bi}, \forall i \in 1, ..., m$. Usually in multiobjective optimization problems, there is a conflict between the objectives: when we want to decrease one,

we increase the other. A solution that is not dominated by any other solution is called *Pareto-optimal*. The set of Pareto-optimal solutions are called *Pareto-optimal set*. See FIGURE 5.

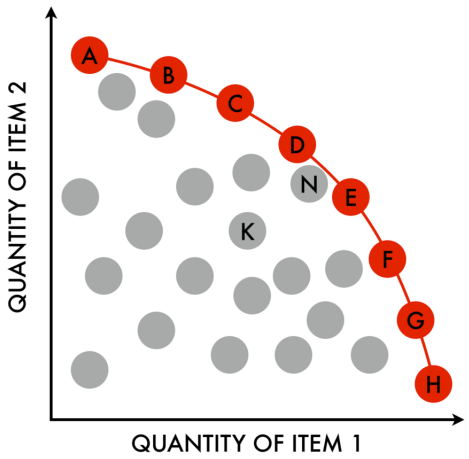

Figure 5: This is an example of two objective optimization, where the target is to maximize both objectives. In this case, points A to H are considered the Pareto optimal solutions. For any of these solutions, you can not get a better solution in any objective without worsening the other objective. Image source is (Wikipedia, 2014)

- **Hypervolume.** On of the critical issues in multiobjective optimization is to assess the quality of the resulting Pareto front and how close the result Pareto front from the true Pareto front. This will also help to compare different multiobjective algorithms. The hypervolume indicator, first appeared in (Zitzler and Thiele, 1998), and further developed in (Zitzler et al., 2007), is one of these measures. See FIGURE 6. It is the size of the space covered by the non-dominated solutions.

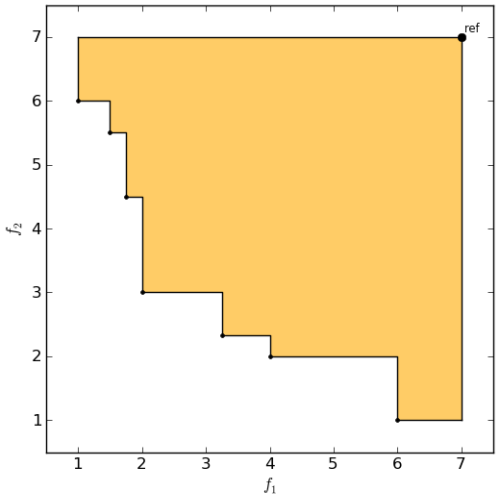

Figure 6: This image is to visualize what is meant by a hypervolume. To calculate the hypervolume, you need to select a reference point. The reference point is a point that is dominated by all other points in the solution space. The reference point is defined by the user, and it depends on the objectives. The colored are in the photo represent the hypervolume in this case. Photo source is (Dortmund, 2011)

- **EHVI (Expected Hypervolume Improvement).** Inspired by the work in (Jones et al., 1998), and a previous work by the same authors in (Tesch et al., 2011),the authors in (Tesch et al., 2013) suggested a new approach for multiobjective optimization. In order to identify the next promising point to try on the robot, the authors select it based on how much improvement this point will contribute to the increase in the hypervolume. The point is selected by optimizing the surrogate functions (an inner optimization for the Gaussian Process for each objective) in order to select this point, and it is up to the researchers to use their favourite built-in optimization methods (in our case, our default optimization consists of iterating over *CMA-ES* (Auger and Hansen, 2005) algorithm two times, and over *NSGA-II* algorithm once. Then, we choose the point that achieves the highest expected hypervolume out of these iterations). The authors report that this approach resulted in significantly reducing the number of trials needed on the robot for finding a good Pareto front

for their objectives. One of the big issues with the *EHVI* is that former techniques available to calculate it used to take considerable computation time. Recent advances in the techniques of calculating the *EHVI* (Hupkens et al., 2014) encouraged its wide usage.

3 Experiments and Results

In order to combine the researches mentioned in the current state of the art, a series of initial experiments have been planned in order to verify the published results so far, understand the various algorithms and the effects of different parameters, and to develop the necessary techniques to combine all these ideas.

3.1 Comparing different multiobjective optimization algorithms using a benchmark

This experiment has been set in order to compare the different algorithms (*NSGA-II* (Deb et al., 2002), *ParEGO* (Knowles, 2006) and *EHVI* (Tesch et al., 2013)) on a benchmark of scalable test problems, developed in (Deb et al., 2005). The advantage of the this benchmark is that it's widely used in comparing multiobjective optimization algorithms, and the optimal Pareto front for it is known. The framework used in order run this is experiment is LIMBO [1]. The experiment setup is as follows:

- Benchmark Problems used: ZDT1 and ZDT2.
  - ZDT1 problem formulation:

$$f_1 = x_1$$
$$f_2 = g.\left(1.0 - \sqrt{f_1/g}\right)$$
$$g(x_2, ..., x_n) = 1.0 + \frac{9}{n-1}\left(\sum_{i=2}^{n} x_i\right)$$
$$0 \leq x_i \leq 1, i = 1, ..., n$$

  - ZDT2 problem formulation:

$$f_1 = x_1$$
$$f_2 = g(\overrightarrow{x}).\left[1.0 - \left(\frac{x_1}{g(\overrightarrow{x})}\right)^2\right]$$
$$g(\overrightarrow{x}) = 1 + \frac{9}{n-1}\left(\sum_{i=2}^{n} x_i\right)$$
$$0 \leq x_i \leq 1, i = 1, ..., n$$

[1] See `https://github.com/jbmouret/limbo.git`

- Total number of evaluations allowed for each algorithm: 100.
- Number of test repetitions (for statistical analysis): 20.
- Comparison between:
  - NSGA-II, with 100 individuals and 30 generations (that is 3000 evaluations, for the sake of comparison only).
  - NSGA-II, with 25 individuals and 4 generations.
  - NSGA-II with 100 individuals and only 1 generation.
  - ParEGO.
  - EHVI with reference point $(11, 11)$.

The comparison of the hypervolume for different algorithms is shown in FIGURE 7 and FIGURE 8.

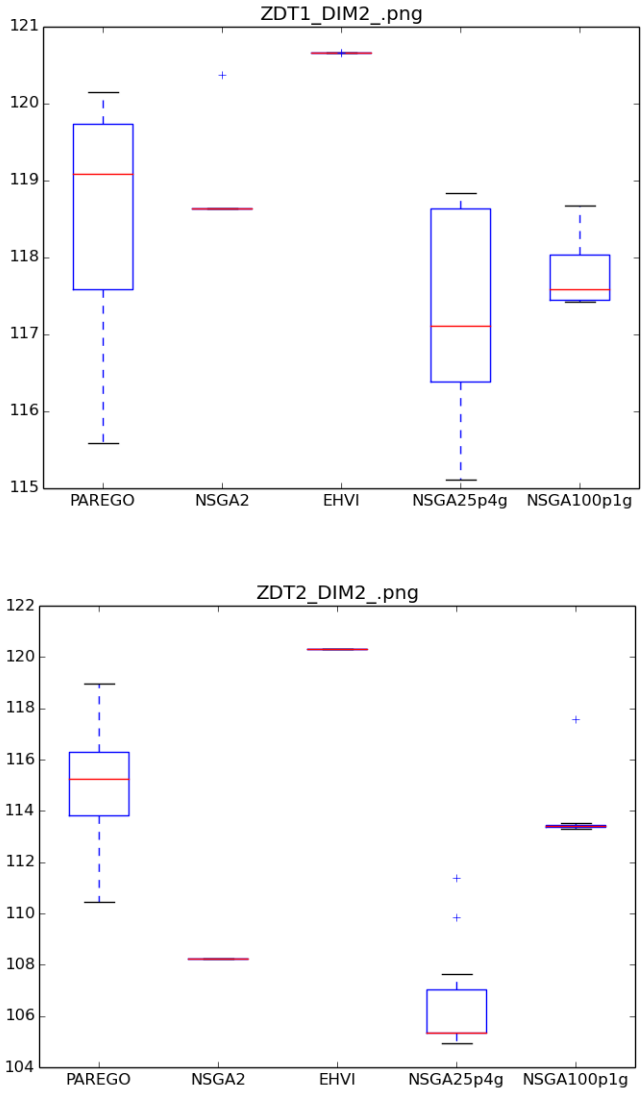

Figure 7: Comparing the hypervolume of the different algorithms for the ZDT1 and ZDT2 benchmark problems, with *2 dimensions*. y-axis is the hypervolume indicator measurement.

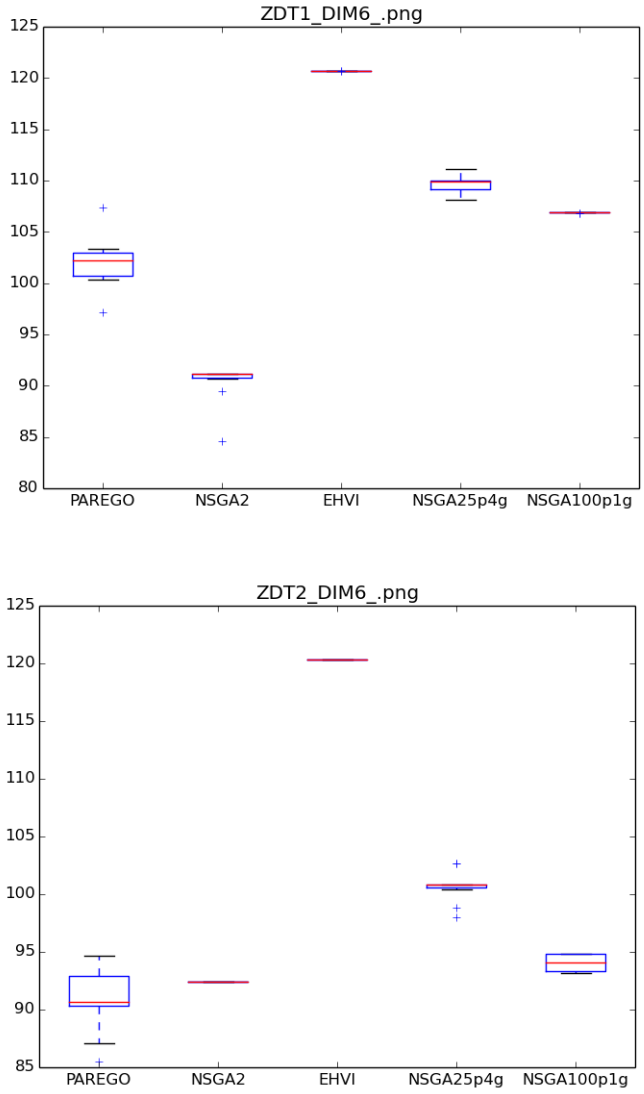

Figure 8: Comparing the hypervolume of the different algorithms for the ZDT1 and ZDT2 benchmark problems, with *6 dimensions*. y-axis is the hypervolume indicator measurement.

It can be seen that *EHVI* has achieved superior performance against all other algorithms (even when we allow NSGA-II to have up to 3000 evalu-

ations). *Wilcoxon Statistical Test* is being used to determine the significance of the difference between different algorithms. Comparing *EHVI* with *NSGA-II* with 25 population and 5 generations in the case of ZDT2 problem with 6 dimension, the result of the statistical test is **0.00506**, which is significant, with *EHVI* having a median hypervolume of **120.32**, while *NSGA-II* having a median hypervolume of **100.85**.

3.2 Comparing different multiobjective optimization algorithms on a robot simulation

In order to further investigate the performance of the different algorithm, we decided to perform another experiment with a more complex experiment on a simulated robot on V-REP Robot Simulation Framework (E. Rohmer, 2013), FIGURE 9.

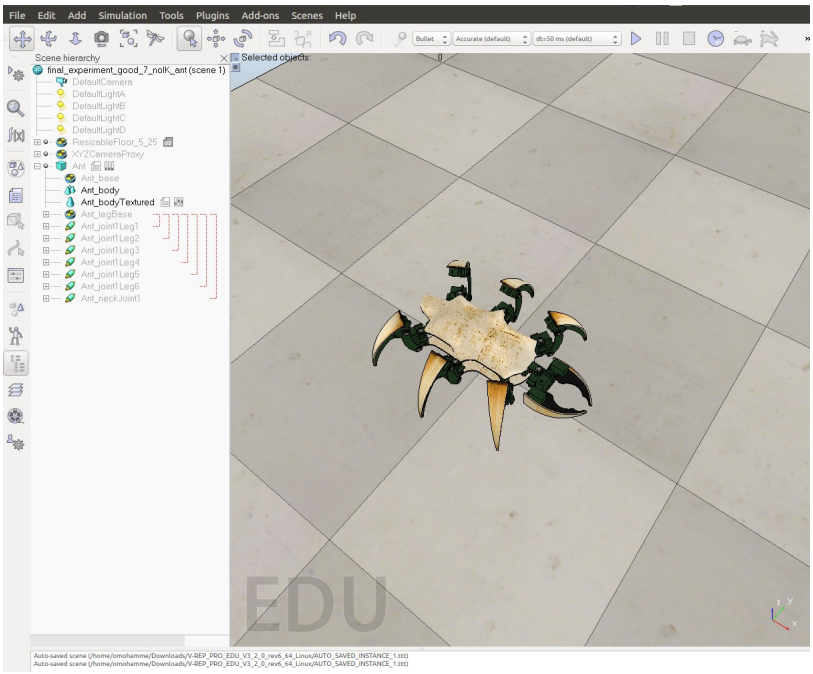

Figure 9: A view of the simulation environment (V-REP). A built-in robot model (the ant model) has been used from the simulator, since it is very similar to the real hexapod robot.

The experiment target is to compare the different state of the art multiobjective optimization algorithms on a more complex scenario, and see how these algorithms perform. To test this, a hexapod robot model has been selected, and the objectives chosen for the algorithms to optimize for are:

- Maximizing the robot speed in a straight line.
- Minimizing the robot base height variations (variations around the z-axis).

There are two main issues that we had to consider while planning for this experiment:

- The two objective should be conflicting with each other. If the two objectives at the end can be reduced to the one objectives, it will be very easy for the algorithms to optimize, and we no longer will have a challenging experiment to test the different optimization algorithms. In the same time, the objective should be feasible to measure on the simulation. For example, a pair of objectives that had been considered was maximizing the robot speed while minimizing the energy consumption of the robot. However, there was some doubts about measuring the robot energy consumption on the simulator, so it was decided to leave such target to be tested on the actual robot.
- The choice of the robot controller - that we want to learn its parameters - plays a role on the novel of the solutions produced. Complex controllers like neural networks can result in new and novel solutions. However, it may be argued that some optimization algorithms can work better on different types of controllers. We decided to make a simple controller, where each joint position can be described as:

$$P_i = \sin\left(time + \phi_i\right) \tag{1}$$

  where P_i the position of joint i. $time$ is the current simulation time, and ϕ_i is the phase shift for this joint. The phase shift for each joint is what we want to learn. In this robot model, we have 18 joints, 3 for each leg. In order to simplify the learning problem, we fix the phases for 6 joints, and learn the phases for the other 12 joints.

The first run of the experiment indicated unexpected bad performance for the *EHVI* algorithm compared to the *NSGA-II* algorithm, which surprised

us. We tried many different configurations related to *EHVI*. The results can be shown in FIGURE 10. The default *EHVI* - *ehvi* - configuration (with reference point $(2, 2)$ in this experiment) used surprisingly bad against *NSGA-II* algorithm, and more notably, against a random choice of parameters - *random*. Several configurations has been tried for the *EHVI* in order to investigate this problem. *ehvi15* and *ehvi11* are where we change the ref point used to calculate the expected hypervolume improvement to $(15, 15)$ and $(11, 11)$ respectively. *ehvi10cmaes* is where I re-run the *CMA-ES* optimization algorithm 10 times, and take its best result. *ehvi10nsgaii* is where I re-run the *NSGA-II* optimization algorithm 10 times, and take its best result. *ehvi100gp* is where the Gaussian Process initialization is repeated 100 times, and the best Gaussian Process is selected from them. *ehvistsa* is where the sampling distribution of the initialization points for the Gaussian Process is changed from uniform sampling to a stratified sampling. The last configuration seems to give better results against most other *EHVI* configurations, yet more investigation for these results are needed.

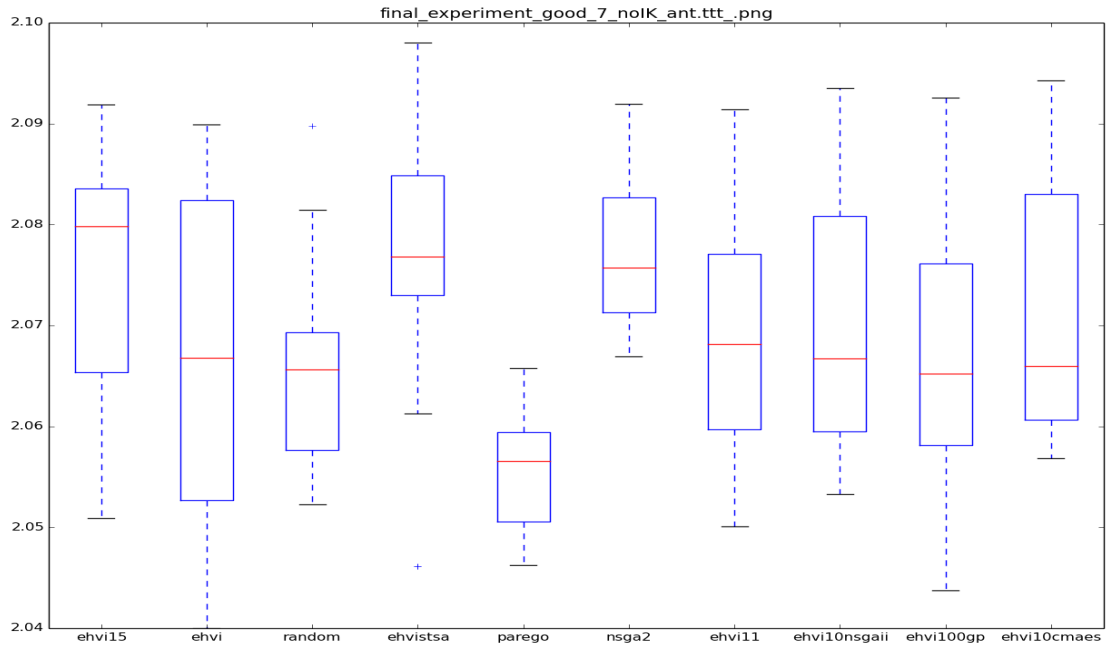

Figure 10: Results of the second experiment. Several configurations have been tried for *EHVI*, without a decisive conclusion.

4 Future Work

As can been in the second experiment, more work needs to be done in order to understand why *EHVI* did not perform well, which is considered to be a smart algorithm, and the team that developed it (Tesch et al., 2013) reported higher performance results than ours. More testing is currently being done about this issue. The latest results indicated that the Gaussian Process used are poorly representing their objective functions. The direct result of this will be of course poor performance for the *EHVI* algorithm. The increase in the problem dimensionality seems to have a direct effect on the performance of the Gaussian Process (in the benchmark, we tested the algorithms against 2D and 6D problems. In the simulator, we working with a 12D problem).

At the current moment, we are getting more evidence that the sampling method used to initialize the Gaussian Process have a direct effect on the Gaussian Process having a meaningful sample to represent the objective function. What was used in all the previous experiment was a simple uniform random sampling from the input space to get the initialization points for the Gaussian Process. However, as the dimensionality goes up, it seems that this sampling method performs poorly. *Stratified Sampling* and *Latin Hypercube Sampling* methods have been implemented to test them in our case, see (Jones et al., 1998), (Sóbester et al., 2005), (McKay et al., 1979).

After resolving this issue, more experiments needs to be done in order to reach the final target of this internship - to be enable the robot to learn quickly to find the set of Pareto front solution to enable it to recover from damage:

- **Measuring the best algorithm performance on the robot.** The physical robot constitutes a more challenging environment, where new elements are introduced, like noise and a more detailed physical environment. This makes the objective functions more difficult to optimize. Adapting the algorithm to this environment, and tuning its parameters correctly is an essential step in order to move forward.
- **Initializing the best algorithm with a prior from simulation and measure performance on the robot.** Once we have all the previous steps have been done, the interesting part here will be to do something like what has been done on (Cully et al., 2015), which is to initialize the

optimization process with a prior made in simulation. It is expected that the robot will be able to find a significantly higher quality (with high hypervolume value) Pareto front in a significantly less number of trials on the robot.

References

A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1769–1776 Vol. 2, Sept 2005. doi: 10.1109/CEC.2005.1554902.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Mahadevan Sridhar (Eds.) Connell, J. H. In J. Fagerberg, D.C. Mowery, and R.R. Nelson, editors, *Robot Learning*. Springer US, 1993.

A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521:503–507, 2015. doi: 10.1038/nature14422.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. In A. Abraham, R. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, chapter 6, pages 105–145. Springer, 2005. ISBN 1-85233-787-7.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

TU Dortmund. Hypervolume — university of tu dortmund, 2011. URL `https://ls11-www.cs.uni-dortmund.de/rudolph/hypervolume/start`. [Online; accessed 08-June-2011].

M. Freese E. Rohmer, S. P. N. Singh. V-rep: a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

K. Goldberg and B. Chen. Collaborative control of robot motion: robustness to error. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 655–660 vol.2, 2001. doi: 10.1109/IROS.2001.976244.

Iris Hupkens, Michael Emmerich, and André Deutz. Faster computation of expected hypervolume improvement. *arXiv preprint arXiv:1408.7114*, 2014.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Joshua Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *Evolutionary Computation, IEEE Transactions on*, 10(1):50–66, 2006.

S. Koos, A. Cully, and J.-B. Mouret. Fast damage recovery in robotics with the t-resilience algorithm. *International Journal of Robotics Research*, 32(14): 1700–1723, 2013. doi: 10.1177/0278364913499192.

Israel Koren and C. Mani Krishna. *Fault-Tolerant Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007. ISBN 0120885255, 9780080492681.

Chih-Min Lin and Chiu-Hsiung Chen. Robust fault-tolerant control for a biped robot using a recurrent cerebellar model articulation controller. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(1): 110–123, 2007.

Michael D McKay, Richard J Beckman, and William J Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

Williams C. K. I. Rasmussen, C. A. MIT Press, 2006.

András Sóbester, Stephen J Leary, and Andy J Keane. On the design of optimization strategies based on global response surface approximation models. *Journal of Global Optimization*, 33(1):31–59, 2005.

M. Tesch, J. Schneider, and H. Choset. Expensive multiobjective optimization for robotics. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 973–980, May 2013. doi: 10.1109/ICRA.2013.6630691.

Matthew Tesch, Jeff Schneider, and Howie Choset. Using response surfaces and expected improvement to optimize snake robot gait parameters. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1069–1074. IEEE, 2011.

Monica L Visinsky, Joseph R Cavallaro, and Ian D Walker. Robotic fault detection and fault tolerance: A survey. *Reliability Engineering & System Safety*, 46(2):139–158, 1994.

Wikipedia. Pareto efficiency — Wikipedia, the free encyclopedia, 2014. URL https://en.wikipedia.org/wiki/File:PareoEfficientFrontier1024x1024.png. [Online; accessed 19-February-2014].

Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In AgostonE. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-65078-2. doi: 10.1007/BFb0056872. URL http://dx.doi.org/10.1007/BFb0056872.

Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Evolutionary multi-criterion optimization*, pages 862–876. Springer, 2007.