FOURIER FEATURES IN REINFORCEMENT LEARNING : AN OVERVIEW OF INPUT PRE-PROCESSING METHODS USING FOURIER FEATURES IN REINFORCEMENT LEARNING

Chupin Clément
Student in Master degree
University of Clermont-Ferrand
clement.chupin.si@gmail.com

Zhang Zhongkai
Chargé de Recherche
INRAE Clermont-Ferrand
zhongkai.zhang@inrae.fr

August 22, 2022

ABSTRACT

In reinforcement learning, understanding the observation space is a key point for all algorithms, and preprocessing the input can improve the convergence and the final performance of the algorithm. Fourier features seem to be a good solution for improving algorithms in RL tasks, in both terms of rewards and sample efficiency, but its use is still recent and applied to small proportion of tasks. Moreover, real-time applications have specific requirements that can be satisfied by Fourier features, such as sample efficiency or better generalization, but the policy and computational limit may be imposed by the task and FFs may become unnecessary or impossible. The use of FF is a bit empirical, there is not enough search around to have a global solution. In this paper, we try to determinate the advantages and disadvantages of the usage of FFs, we present multiple input preprocessing experiments, on a multilayer perceptron (MLP), we aim to exploit most variations of Fourier features, applied to most classical policies (on/off policy, continuous and discrete action space), to most tasks (mujoco, classical open ai gym control). To the best of our knowledge, this is the first global comparative evaluation of Fourier features in deep RL. Fourier features in deep RL. Finally, we propose a standard input processing of Fourier features that best combines the advantages of these techniques.

1 Introduction

Recent works has shown than Random Fourier Features can increase the dimensionnal interpretaion of a neural network (Tancik et al., 2020 [1]). And more specific in Reinforcement Learning, the RFF (Random Fourier Features) of the observation space shown an improvement in convergence steps (Brellmann et al., 2021 [2]), but requires more computational power for each step, the global improvement have to be evaluate. This work tend to complete these studies to expand the use of Fourier features in RL task. Others works has shown an interest in using other variations of the Fourier Features such as periodic activation function in a classical neural network (Sitzmann et al.,2019 [3]) and the application of this method in the domain of RL, at our knowledge, has not been tested in the past. An other point interesting in robotics and real-time application of FFs is the sample effiency, recording of a large enough dataset can be difficult and expensive. And in despite of the implementation and the computional power, the FFs is a good improvement for this domains. This paper tries to determine the advantages and disadvantages of FFs compared to classical MLP (no input preprocessing), with a evaluation of the difference in reward and computational power usage, most of our work was to determine the best variants of each FF, but we found out that there is no standard way to deal with them, we have to try the best pair for each problem and policy. Regarding real time, with this study we hope to bring RL algorithms into new real time tasks, especially in robotics, which require good sampling efficiency and generalization with limited hardware. Due to the cost of sample acquisition, safety and robustness of classical automatic engineering, the use of neural networks in direct control is not yet common, FFs can be a good point to facilitate the introduction of RLs in real life.

2 Background

2.1 Global Neural Network Processing

The Fourier Features is a transformation of the input to have a better representation of the input for the neural network, the input is convert in a biggest expression space, by multi-frequency representation of each input.

You can see in details the matrix processing here : `https://github.com/clement-chupin/BenchNeuralNework/blob/master/utils_lib/feature_extractor_layers.py`

Notation :

$I \in \mathbb{R}^n$: Input matrix of dimension

$X \in [0,1]^n$: Input matrix normalized

$K \in \mathbb{R}^{i,j}$: Features Feature kernel

$H \in \mathbb{R}^{k,l}$: Result of the multiply of the input by the kernel

$Z \in \mathbb{R}^p$: Features Extracted matrix, periodic and flatten version of H

$Y \in \mathbb{R}^a$: Output matrix

$o \in [1, \sim 64]$: order of FFs (by F))

$\mathbb{R}^n \to \omega(X) \to [-1,1]^n$: Periodic method ($\cos(X)$ by C)

$\mathbb{R}^n \to \psi \to [0,1]^n$: Normalization method ($[0,1]$ by C)

2.1.1 Overview of the global network processing

$I = Input$

$X = Normalize(I)$

$H = X * Kernel * 2\pi$

$Z = FeatureExtraction(H)$

$Y = NeuralNetwork(Z)$

$Output = Y$

The method called $NeuralNetwork$ is the RL agent process, for specific policy, we can have different action architecture, based on the feature extraction (critic/actor neural network for example).

Notation of a classical FF, the normalized input is multiply by the kernel, to be process by a periodic function ($input * kernel = output$) :

$$\omega(\begin{bmatrix} input_{(0)} \\ input_{(1)} \\ ... \\ input_{(n)} \end{bmatrix} * \begin{bmatrix} kernel_{0,0} & kernel_{0,1} & ... & kernel_{0,j} \\ kernel_{1,0} & kernel_{1,1} & ... & kernel_{1,j} \\ ... & ... & ... & ... \\ kernel_{i,0} & kernel_{i,1} & ... & kernel_{i,j} \end{bmatrix}) = \begin{bmatrix} output_0 \\ output_1 \\ ... \\ output_p \end{bmatrix} \tag{1}$$

2.2 Fourier Feature / Fourier Light Feature

There are two ways to process an Fourier Feature extraction, the Fourier Feature (FF) and the Fourier Light Feature (FLF), all Fourier Feature and their variants are note FFs. This technics are taken from (Tancik et al., 2020) and (Brellmann et al., 2021). In this paper, we aim to have a explicit and the simpliest explanation of FFs.

2.2.1 FF : Fourier Feature

This method convert the input to bigger observation space to have more information. With respect of a Fourier Transform, the input are mixed.

In dimensions space, we have : $X^n \to Z^p$, the p is dependant of the order and of the variant of the FF.

The simplified formula is : $FF_i(X) = \omega((X_0 * K_{0,i} + X_1 * K_{1,i} + ... + X_n * K_{n,i}) * 2\pi)$ with $i \in [0, p]$

The final equation is :

$$Z_i = FF_i(X) = \omega(\psi(I)^\intercal * K * 2\pi) = \sum_{j=0}^{n} \psi(X_j) * K_{j,i} * 2\pi \tag{2}$$

with $shape(K) = (n, p,) \to K^{n*p}$ we have :

$$\omega(I^{1,n} * K^{n,p} * 2\pi) = \omega(H^{1,p}) = Z^p \tag{3}$$

2.2.2 FLF : Fourier Light Feature

This method is a light version of the FF, but with benefits, the output space is proportional to the input space and the order. Each input is decomposed in different periodic reading of the value.

In dimensions space, we have : $X^n \to Z^p$ with $p = n * order$

The final equation with $i \in [0, n]$ and $j \in [0, order]$ is :

$$FLF_{i,j}(X) = \omega((\psi(I_i) * K_j) * 2\pi) \tag{4}$$

with $dim(K) = (1, o,) \to K^{1,o}$ we have :

$$\omega(X^{n,1} * K^{1,o} * 2\pi) = \omega(H^{n,o}) = Z^p \tag{5}$$

2.3 Deterministic / Random / Learned Fourier Feature

The main part of the the FFs is the kernel value, they can be determinate, random or learned, this variations cause several changes in the feature extraction processing. The notation is DFF/DFLF for Deterministic Fourier Feature or Fourier Ligth Feature, RFF/RFLF for Random FF and FLF, and LFF/LFLF for Learned FF and FLF.

2.4 Deterministic

The Deterministic Fourier Feature and the Deterministic Fourier Light Feature are input pre processing method who have explicit transformation based on explicit equation witout weight regularization or random value.

2.4.1 DFF

: Deterministic Fourier Feature let K be determinate by classical Fourier Feature transform. It's a interressant and an explicit technics, each combination of each input is present, but it become impossible when the observation space is too big (Impossible pre-processing : IPP).

With $X^n \to Z^p, p = o^n$

with $i \in [0, n]$, $j \in [0, p]$ and $dim(K) = (n, p,) \to K^{n*p}$:

$$K_{i,j} = int(j/o^i) \mod o \tag{6}$$

with int() round to the lower integer

This explain the impossible pre-processing for big input, for example, to resolve the mujoco problem Ant-v4 from [4]

We have as input $n = 27$, we choose a litle order like $o = 2$, we finaly have $p = 2^{27} = 134.217.728$ who is hard to compute and useless to achieve robotics problems.

2.4.2 DFLF

: Deterministic Fourier Light Feature let K be determinate by classical simplified Fourier Feature coeff
with $i \in [0, o]$ and $dim(K) = (1, o,) \rightarrow K^{1*o}$:

$$K_{1,i} = i + 1 \tag{7}$$

2.5 Random

2.5.1 RFF

: Random Fourier Feature let K with random normalized coeff
with $i \in [0, n]$, $j \in [0, p]$ and $dim(K) = (n, p,) \rightarrow K^{n*p}$:

$$K_{i,j} = r \in [0, 1] \tag{8}$$

2.5.2 RFLF

: Random Fourier Light Feature let K with random normalized coeff
with $i \in [1, order + 1]$ and $dim(K) = (o,) \rightarrow K^{o}$:

$$K_{i} = r \in [0, 1] \tag{9}$$

2.6 Learned

LFF and LFLF, Learned Fourier Feature and Learned Fourier Light Feature let K be find by the neural network by basic SGD. The LFF and LFLF is close to the SIREN (Sitzmann et al., 2020 [3]), a neural network with sinusoidal activation function.

2.6.1 FF

: represented by : $SIREN(X) = \omega(X * K)$ X*K is classical full connected layer :
with $dim(K) = (n, p) \rightarrow K^{n*p}$:

$$\omega(X^{1*n} * K^{n*p} * 2\pi) = \omega(H^{1*p}) = Z^{p} \tag{10}$$

2.6.2 FLF

: represented by : $SIREN_L(X) = \omega(X * K)$ X*K is classical full connected layer :
with $dim(K) = (1, o) \rightarrow K_{1*o}$:

$$\omega(X^{n*1} * K^{1*o} * 2\pi) = \omega(H^{n*o}) = Z^{p} \tag{11}$$

3 Result (in construction)

3.1 Final performance

First thing we will benchmark is the final performance by the reward mesure between the classical and the FF, we normalize by the classical version.

3.2 Convergence gains

Second mesure is design for robotics task, it will mesure how faster the FFs will reach similar performance of the classical version.

3.3 CPU power consumption

We will mesure the speed cost of the FFs in comparaison to the classical version, the performance mesure is from the trainning and the execution time.

3.4 Total gains

The previous section show us a decrease in the convergence time, but increase in computation time, now, we have the final performance mesure for the training time.

4 Conclusion (in construction)

4.1 Global view of FFs

4.2 FFs for robotics : pros and cons

4.3 New applications of FFs

References

[1] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *CoRR*, abs/2006.10739, 2020.

[2] David Brellmann, Goran Frehse, and David Filliat. Fourier features in reinforcement learning with neural networks, 2022.

[3] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[5] Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

[6] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

A Relative study

By our goal and our computation limit, we have decide to have a global overview on our algorithms. The main purpose is to evaluate the difference between the classic version of MLP and the version with Fourier pre-processing. According with Open AI zoo [5] and our goals, we have decide to evaluate environements on only one billions of timesteps. We might evaluate on more but there is not a big impact on the final rewards, and our study is focus on the sample efficiency gains of complex problems with limited training time.

B Normalization

The Fourier Feature need a input in a closed range, we have decide to use the simpliest way to normalize it, by the different task, and to don't affect the different FFs, we have decide to not paramterize activation function to feat to the input to normalize it (especially tanh function). We have used an VecNormalize from [6].

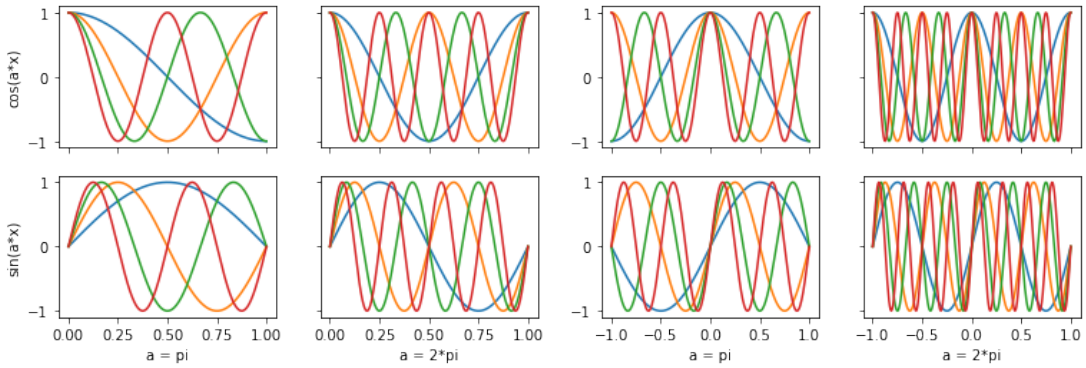
The function $\psi_{0,1}(I) \in \mathbb{R}$ will convert the input to normalize it : $\mathbb{R}^n \to [0,1]^n$. It's a simple floating window taken from Open AI (stable-baseline3), the equation behind is :

$$\psi_{a,b}(I) = (a+b)/2 + \frac{I - mean(I)}{(b-a) * \sigma(I)} \tag{12}$$

In the paper, we assume that $\psi(I) = \psi_{0,1}(I) = X$, but another normalization for another search is possible.

C Observation range and periodic function

We have test some other couple of range/periodic function who are all similar, but with default for some policie with some feature extraction method. According to our experiments and the others studies, we have decide to use $\cos(X), X \in [0, \pi]$ observation range.



Range and periodic function search by plotting

$\cos(x), x \in [0, \pi]$: perfect for our search

$\sin(x), x \in [0, \pi]$: dead zone in 0 and pi

$\cos(x), x \in [0, 2 * \pi]$: dead zone in 0 and 2*pi, symetric

$\sin(x), x \in [0, 2 * \pi]$: dead zone in 0, pi and 2*pi

$\cos(x), x \in [-\pi, \pi]$: symetric

$\sin(x), x \in [-\pi, \pi]$: dead zone in -pi, 0 and pi

$\cos(x), x \in [-2 * \pi, 2 * \pi]$: symetric, repetition

$\sin(x), x \in [-2 * \pi, 2 * \pi]$: dead zone in -2*pi, 0 and 2*pi, repetition

Dead zone for point where all frequency go to the same value, on two or more zone, for the neural network, there is no difference between two different dead zone.

Concerning the periodic function, according to theoretical and experimentation, and following [2], we present $\cos(X)$ as the best periodic function to our study case.

Concerning the normalization range, in despite of robotics recommendation of a range $[-1, 1]$, to the same reason of the periodic function choice, we present the range $[0, 1]$ as the best range to our study case.

D Logging

At the beginning of the studie, we wanted to register all data, the replay buffer, the final trained model, but there was too many of experiment to store it all, and no interest to a human to see the final resolution of the problem, we can't compare a the naked eye which one is better of an other.

To have a flexible execution, we have all the log in different files, with dynamic write, we might use SQL database for a better comparaison but with the execution on an other computer, the manipulation of simple files is easier.

E Neural Network

We have used the Open AI stuff to our model and our policies, the purpose of this choice is to test the most popular implementation of policies, with the most basic and common neural network behind the feature extraction, we don't touch to the neural architeture (all policie have different architecture and sometime multiple neural network....),

F Hyperparameters of policies

Table 1: A2C

HP	Value
Neural Network	[Action, Value]
Number of Hidden Layers	[2,2]
Number of Neurons per Hidden Layer	[[64,64],[64,64]]
Activation function	[tanh,tanh]
Discount Factor (gamma)	0.99
Learning rate	0.0007
Fourier Order (FF)	1, 2, 3, 4, 5
Fourier Light Order (FLF)	1, 2, 3, 4, 5

Table 2: DDPG

HP	Value
Neural Network	[Actor, Actor Target, Critic, Critic Target]
Number of Hidden Layers	[2,2,2,2]
Number of Neurons per Hidden Layer	[[400,300],[400,300],[400,300],[400,300]]
Activation function	[relu,relu,relu,relu]
Discount Factor (gamma)	0.99
Soft Update Coefficient (tau)	0.005
Learning rate	0.001
Fourier Order (FF)	1, 2, 3, 4, 5
Fourier Light Order (FLF)	1, 2, 3, 4, 5

Table 3: DQN

HP	Value
Neural Network	[Q Net, Q Net Target]
Number of Hidden Layers	[2,2]
Number of Neurons per Hidden Layer	[[64,64],[64,64]]
Activation function	[relu,relu]
Discount Factor (gamma)	0.99
Soft Update Coefficient (tau)	1
Exploration fraction (epsilon)	auto in [0.1,1]
Learning rate	0.0001
Fourier Order (FF)	1, 2, 3, 4, 5
Fourier Light Order (FLF)	1, 2, 3, 4, 5

Table 4: PPO

HP	Value
Neural Network	[Action, Value]
Number of Hidden Layers	[2,2]
Number of Neurons per Hidden Layer	[[64,64],[64,64]]
Activation function	[tanh,tanh]
Discount Factor (gamma)	0.99
Batch size	64
Learning rate	0.0003
Fourier Order (FF)	1, 2, 3, 4, 5
Fourier Light Order (FLF)	1, 2, 3, 4, 5

Table 5: SAC

HP	Value
Neural Network	[Actor, Critic, Critic Target]
Number of Hidden Layers	[2,2,2]
Number of Neurons per Hidden Layer	[[256,256],[256,256],[256,256]]
Activation function	[relu,relu,relu]
Discount Factor (gamma)	0.99
Soft Update Coefficient (tau)	0.005
Batch size	64
Learning rate	0.0003
Fourier Order (FF)	1, 2, 3, 4, 5
Fourier Light Order (FLF)	1, 2, 3, 4, 5

Table 6: TD3

HP	Value
Neural Network	[Actor, Actor Target, Critic, Critic Target]
Number of Hidden Layers	[2,2,2,2]
Number of Neurons per Hidden Layer	[[400,300],[400,300],[400,300],[400,300]]
Activation function	[relu,relu,relu,relu]
Discount Factor (gamma)	0.99
Soft Update Coefficient (tau)	0.005
Learning rate	0.001
Fourier Order (FF)	1, 2, 3, 4, 5
Fourier Light Order (FLF)	1, 2, 3, 4, 5

G Open problems

To compete with rigor [2], we could have bench :

HP robustness : vari the learning rate, the policy parameters and compare the reaction of the FFs, maybe some parameters can totaly break the FF.

But by the limitations in computation, time, general knowledge and finaly, the way of this study, we have decide to not deepen this different specific task to each policy.

To compete with rigor of [[2], [1]], we have to so study the kernel regression repartition of the RFF / RFLF to improve this technics in RL domain. Maybe combine a DFLF with noise can bring solutions.

An other point of interest is the use of Learned FF, in this study, there is no deep look up on this technics and the way to evolve the kernel, may be use an other neural network who is trained to choose the best kernel for a specific task, but, like the AutoML, the meta-learning is difficult to have an interest (to have a better convergence, we have to launch the trainning a lot to learn the FF, the interest is limited).

H special thanks

Thanks to David Brellmann to his collaboration and his DFF_LL and DFLF_LL extraction layer code.

Thanks to the INRAE Clermont-Ferrand to the welcome at my internship Thanks to Mésocentre Clermont Auvergne for his supercomputer to facilities of the computation

Table 7: Computation time

All params		Result					
Env	metrics	A2C	DDPG	DQN	PPO	SAC	TD3
ant	none	384.0	247.0	NC	480.0	16.0	243.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	49.0	228.0	NC	332.0	11.0	225.0
	dflf	57.0	222.0	NC	68.0	12.0	223.0
	rff	234.0	236.0	NC	475.0	15.0	229.0
	rflf	58.0	224.0	NC	69.0	11.0	222.0
	lff	243.0	228.0	NC	473.0	14.0	224.0
	lflf	51.0	205.0	NC	327.0	8.0	203.0
halfcheetah	none	974.0	496.0	NC	1378.0	20.0	490.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	104.0	353.0	NC	927.0	15.0	329.0
	dflf	150.0	309.0	NC	259.0	15.0	310.0
	rff	543.0	375.0	NC	1340.0	18.0	383.0
	rflf	150.0	320.0	NC	259.0	15.0	309.0
	lff	522.0	316.0	NC	1304.0	16.0	309.0
	lflf	95.0	287.0	NC	881.0	11.0	283.0
hopper	none	686.0	77.0	NC	853.0	19.0	232.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	273.0	164.0	NC	575.0	16.0	100.0
	dflf	78.0	72.0	NC	571.0	16.0	119.0
	rff	381.0	34.0	NC	822.0	17.0	57.0
	rflf	76.0	116.0	NC	578.0	16.0	118.0
	lff	372.0	171.0	NC	822.0	15.0	138.0
	lflf	246.0	127.0	NC	560.0	11.0	53.0
humanoidstandup	none	52.0	153.0	NC	362.0	11.0	158.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	22.0	108.0	NC	36.0	2.0	104.0
	dflf	30.0	114.0	NC	41.0	2.0	116.0
	rff	47.0	154.0	NC	208.0	13.0	150.0
	rflf	28.0	121.0	NC	41.0	2.0	115.0
	lff	45.0	146.0	NC	203.0	11.0	149.0
	lflf	26.0	96.0	NC	39.0	2.0	113.0
humanoid	none	53.0	61.0	NC	428.0	12.0	65.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	29.0	41.0	NC	39.0	2.0	34.0
	dflf	33.0	43.0	NC	46.0	3.0	35.0
	rff	50.0	71.0	NC	233.0	13.0	84.0
	rflf	33.0	69.0	NC	46.0	3.0	46.0
	lff	48.0	85.0	NC	231.0	12.0	93.0
	lflf	28.0	61.0	NC	43.0	2.0	58.0
inverteddoublependulum	none	775.0	125.0	NC	962.0	19.0	47.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	303.0	25.0	NC	667.0	16.0	24.0
	dflf	82.0	25.0	NC	661.0	16.0	25.0
	rff	420.0	33.0	NC	947.0	18.0	31.0
	rflf	79.0	27.0	NC	660.0	16.0	26.0
	lff	406.0	23.0	NC	947.0	16.0	24.0
	lflf	286.0	18.0	NC	651.0	11.0	19.0
invertedpendulum	none	1130.0	671.0	NC	1606.0	20.0	574.0
	dff	106.0	19.0	NC	156.0	14.0	65.0
	dfflfl	764.0	67.0	NC	1575.0	19.0	61.0
	dflf	189.0	122.0	NC	1543.0	19.0	105.0
	rff	606.0	33.0	NC	1566.0	19.0	31.0
	rflf	190.0	52.0	NC	1542.0	19.0	33.0
	lff	584.0	24.0	NC	1553.0	17.0	514.0
	lflf	693.0	21.0	NC	1447.0	13.0	185.0
reacher	none	1060.0	473.0	NC	1467.0	21.0	465.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	414.0	333.0	NC	1005.0	17.0	311.0
	dflf	107.0	298.0	NC	989.0	17.0	277.0
	rff	579.0	394.0	NC	1430.0	18.0	366.0
	rflf	110.0	303.0	NC	1005.0	17.0	261.0
	lff	564.0	259.0	NC	1391.0	17.0	255.0
	lflf	384.0	237.0	NC	937.0	12.0	235.0
swimmer	none	884.0	431.0	NC	1246.0	20.0	451.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	358.0	323.0	NC	869.0	17.0	309.0
	dflf	101.0	308.0	NC	862.0	17.0	308.0
	rff	511.0	350.0	NC	1217.0	18.0	314.0
	rflf	102.0	310.0	NC	866.0	18.0	307.0
	lff	466.0	310.0	NC	1225.0	16.0	302.0
	lflf	326.0	287.0	NC	813.0	12.0	280.0
walker2d	none	667.0	256.0	NC	816.0	19.0	263.0
	dff	IPP	IPP	NC	IPP	IPP	IPP
	dfflfl	72.0	236.0	NC	558.0	14.0	242.0
	dflf	112.0	205.0	NC	160.0	14.0	251.0
	rff	369.0	233.0	NC	790.0	17.0	251.0
	rflf	112.0	217.0	NC	159.0	14.0	238.0
	lff	359.0	180.0	NC	790.0	15.0	208.0
	lflf	68.0	136.0	NC	540.0	10.0	68.0
acrobot	none	1557.0	NC	5219.0	2568.0	NC	NC
	dff	97.0	NC	5252.0	258.0	NC	NC
	dfflfl	1051.0	NC	5262.0	1749.0	NC	NC
	dflf	1046.0	NC	5239.0	1698.0	NC	NC
	rff	1483.0	NC	5278.0	2476.0	NC	NC
	rflf	1033.0	NC	5203.0	1719.0	NC	NC
	lff	1433.0	NC	5236.0	2399.0	NC	NC
	lflf	949.0	NC	5188.0	1583.0	NC	NC
	none	1740.0	NC	7428.0	2967.0	NC	NC