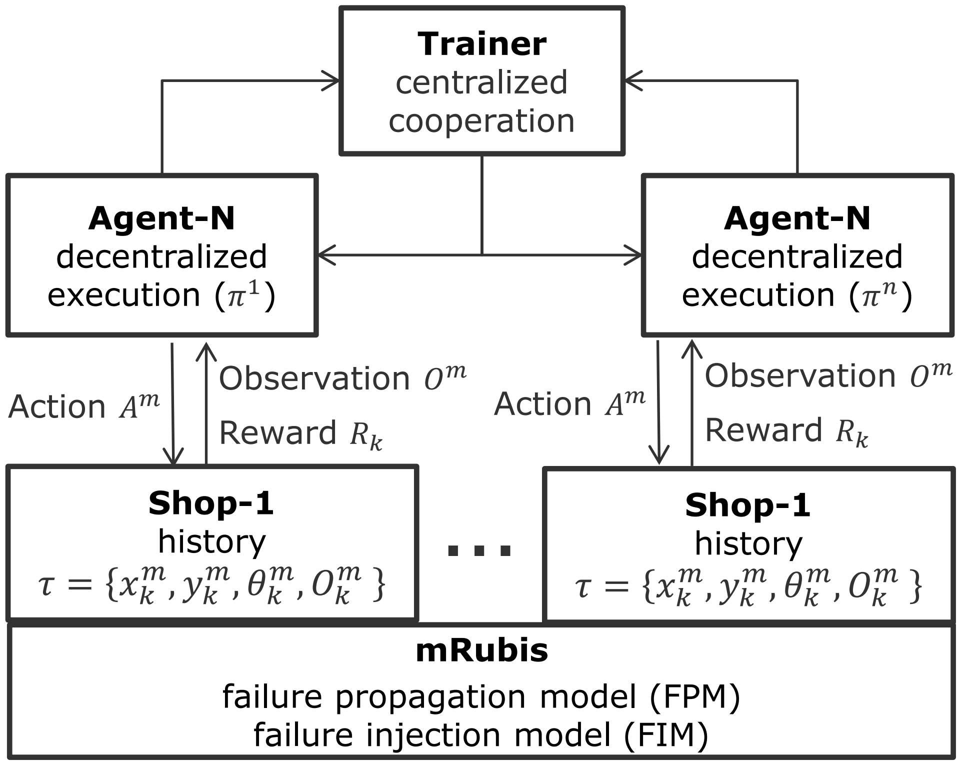
Project Plan

Christian M. Adriano
christian.adriano@hpi.de

Scope

Scenario – Multi-Agent Reinforcement Learning



Definition – Agents e Trainer

- Each agent executes a policy on its local history
- The local history τ consists of sequence of observations O^m, Q-value $Q(O^m, A^k)$, and parameters θ of the policy $\pi_\theta(O^m)$
- Each agent shares history τ with the trainer component, which can later use it to train a new version of the policy $\pi_\theta(O^m)$
- The trainer learns a new policy $\pi'_\theta(O^m)$ and deploy it to the agents

Design Questions

- When and how does each agent share its local history?
- When does the trainer decide that it should learn a new policy?
- When does the agent decide to replace its policy for the new policy?

Definitions – Shops and Components

A shop consists of components (18), which have specific responsibilities and present interdependencies (directed). Components can have multiple instances to support surges in demand.

The utility of a shop consists of the sum of the utility of its components. The utility of a component is a function of its parameters.

The parameters of a component consist of:

Shops work in isolation and do not interfere with the operation of other shops.

Definitions – Failures and Actions

One or more components in a shop can fail at any given time. These failures are call failure modes:

$$CF_0 \rightarrow \{CF_1, .., CF_5\}$$

- CF_1 = Component crashed
- CF_2 = Component throws exceptions
- CF_3 = Component is undeployed
- CF_5 = Change in the system load causing sub-optimal performance

CF1,CF2,CF3 can be addressed by:

- Restart Component
- Heavy Weight Redeployment
- Light Weight Redeployment
- Replace Component (if an alternative component is available)

CF5 can be addressed by:

- Add Replica
- Remove Replica

Possible Actions in mRUBiS

- Repair Action
  - Restart Component
  - Heavy Weight Redeployment
  - Light weight Redeployment
  - Replace Component (if an alternative component is available)

- Optimization Actions
  - Add Replica
  - Remove Replica

Action: <Reward, Cost>
Reward (Affected_Component , CF_type)

Definition
Failure Propagation Pattern

Failure in one component can affect other components, when this happens, we have a failure propagation pattern (FPP), which is a sequence of failure modes and observations $FPP = \{So_0^1, Sd_1^1, Su_1^2, Ou_2\}$

An FPP is generated by Failure Injection Model (FIM)

- Each shop has an instance of the failure propagation model (FPM)
- The FPM is implemented using a Hidden Markov Model, which determines the probability that a component will fail given that other components are also failing.
- Failure patterns (FP) are generated by failure injection event (FIE) given failure propagation model (FPM)

Tasks:

- Standardize the nomenclature
- Map it to mRubis
- Describe how FPM, FIE, and FPP will be reified in mRubis

System Goal

- Keep e-commerce platform operating at maximum utility, which entails having all shops operating at maximum utility
- Maximum utility consists of a system state x_k where all components across all shops are operational, i.e., in CF_0 mode
- This requires that the controllers of the system counteracts the disturbances as quickly as possible to minimize the cumulative drop in total utility = $\sum_{1 \in K} \sum_{i \in \{1,m\}} y^m$, i.e., sum of all individual utilities in K steps

Project Plan

Levels of Robustness

Level 1 – Robustness to Failure Propagation

- Static Confounding (failure masking)
- Dynamic Confounding (intermittent failure)

Level 2 - Robustness to Under-specification

- OOD (Training does not cover the entire distribution space)
- Rashamon sets with the HMMs (Failure Propagation Models are not Unique, Equivalence Classes, but some models present shortcuts, spurious correlations)

Level 3 - Robustness to Distribution Shifts

- Concept drift (Changes in the System Utility – Nonstationarity)
- Risk driven training for risk-averse evolution (safety)

Preliminary Work-Packages

1- Failure Propagation Model (Hidden Markov Model) - Florence

Study how to move the HMM implemented in the Python side to the Java side

2- Failure Injection Mechanism - Christopher

Study how to generate failure injects that reflect that failure propagation patterns

3- Reinforcement Learning-Part-1 Ulrike & Jonas

Study how to replace the Supervised Learning controller (Regression) with a Self-Supervised One (RL)

4- Multi-Agent RL: Study how to have two agents, each responsible for one shop.

5- Monitoring for Transfer Learning: Study how to measure the differences in policies across agents.

6- Robustness Tests: Study how to generate stress tests that show how policies are robust to perturbations, i.e., the agent is still able to fix the failures of its shop(s) with a minimal degradation in utility

Evaluation

1- Value-at-Risk* Trade-offs

Goal: Show different rates of synchronization among Agents:

1.1 excessive cost of training and redeployment

1.2 increase in the risk of under-performance

Cumulative performance

2- Convergence

Goal: Show that different strategies* to learn when to train and redeploy require:

2.1 more data to achieve an average value-at-risk

2.2 longer time to converge

*strategies could be different hyper-parameterizations of the algorithms

Look at visualization of RL outcomes - https://araffin.github.io/post/rliable/

End

Data Used for the Implementation

Optimal_Index	Optimal_Failure	Optimal_Affected_Component_Uid	Optimal_Affected_Component	Optimal_Utility_Drop	Optimal_Rule	Optimal_Utility_Incre	CycleID	CycleSize
1	CF1	_SEu7g-cdEeet0YmmfbMwkw	Item Management Service	2223.302629	RestartComponent	2223.302629	1.2	7
2	CF5	_SEu7jecdEeet0YmmfbMwkw	Bid and Buy Service	1986.620131	AddReplica	3078.948412	1.2	7
3	CF1	_SEuUY-cdEeet0YmmfbMwkw	Past Sales Item Filter	1345.11739	RestartComponent	1345.11739	1.2	7
4	CF2	_SEu7tecdEeet0YmmfbMwkw	Availability Item Filter	1339.211283	RestartComponent	1339.211283	1.2	7
5	CF1	_SEuUO-cdEeet0YmmfbMwkw	Item Management Service	1027.915526	RestartComponent	1027.915526	1.2	7
6	CF2	_SEuUbecdEeet0YmmfbMwkw	Availability Item Filter	620.7804748	RestartComponent	620.7804748	1.2	7
7	CF3	_SEvipucdEeet0YmmfbMwkw	Bid and Buy Service	1598.612985	HwRedeployComponent	1598.612985	1.2	7
1	CF5	_SEvio-cdEeet0YmmfbMwkw	Authentication Service	1152.384931	AddReplica	1837.951446	2.2	9
2	CF5	_SEuUQucdEeet0YmmfbMwkw	Authentication Service	1067.673858	AddReplica	1465.530998	2.2	9
3	CF3	_SEu7iucdEeet0YmmfbMwkw	Authentication Service	3634.777573	ReplaceComponent	4192.495152	2.2	9
4	CF5	_SEwwpOcdEeet0YmmfbMwkw	Authentication Service	447.7781247	AddReplica	834.8188024	2.2	9
5	CF3	_SEx_HucdEeet0YmmfbMwkw	Authentication Service	2418.121625	ReplaceComponent	2789.933345	2.2	9
6	CF5	_SEymC-cdEeet0YmmfbMwkw	Authentication Service	466.255042	AddReplica	622.2518251	2.2	9
7	CF5	_SExYMecdEeet0YmmfbMwkw	Authentication Service	419.9089219	AddReplica	621.3839818	2.2	9
8	CF3	_SExXkecdEeet0YmmfbMwkw	Authentication Service	232.8780733	ReplaceComponent	270.0592453	2.2	9
9	CF3	_SEwJrOcdEeet0YmmfbMwkw	Authentication Service	229.0959886	ReplaceComponent	266.2771606	2.2	9
1	CF5	_SEu7sOcdEeet0YmmfbMwkw	Buy Now Item Filter	1589.786027	AddReplica	3322.913203	3.2	12
2	CF2	_SEviuOcdEeet0YmmfbMwkw	Persistence Service	2154.517826	RestartComponent	2154.517826	3.2	12
3	CF5	_SEu7n-cdEeet0YmmfbMwkw	Persistence Service	1555.14053	AddReplica	3062.343411	3.2	12
4	CF2	_SEviyecdEeet0YmmfbMwkw	Buy Now Item Filter	1806.53671	RestartComponent	1806.53671	3.2	12
5	CF5	_SEu7xecdEeet0YmmfbMwkw	Comment Item Filter	752.3169425	AddReplica	2462.633047	3.2	12
6	CF5	_SEu7uucdEeet0YmmfbMwkw	Region Item Filter	177.709237	AddReplica	2445.591773	3.2	12
7	CF5	_SEu7pecdEeet0YmmfbMwkw	Last Second Sales Item Filter	1860.956568	AddReplica	2197.106169	3.2	12
8	CF2	_SEvi0-cdEeet0YmmfbMwkw	Region Item Filter	1431.510979	RestartComponent	1431.510979	3.2	12

Deliverables

- Learn a model for the system dynamics (only function $\boldsymbol{g_\theta}$) that is a good approximation
- Build a control law (function $\boldsymbol{h}$) that minimizes cumulative utility drop
- Evaluate of these models under various situations (sensitivity analysis)

Trade-off between Sarsa and Q-Learning

Gridworld example from [Sutton & Barto 2018] page 132.

Goal: is to go from S to G avoiding the Cliff.