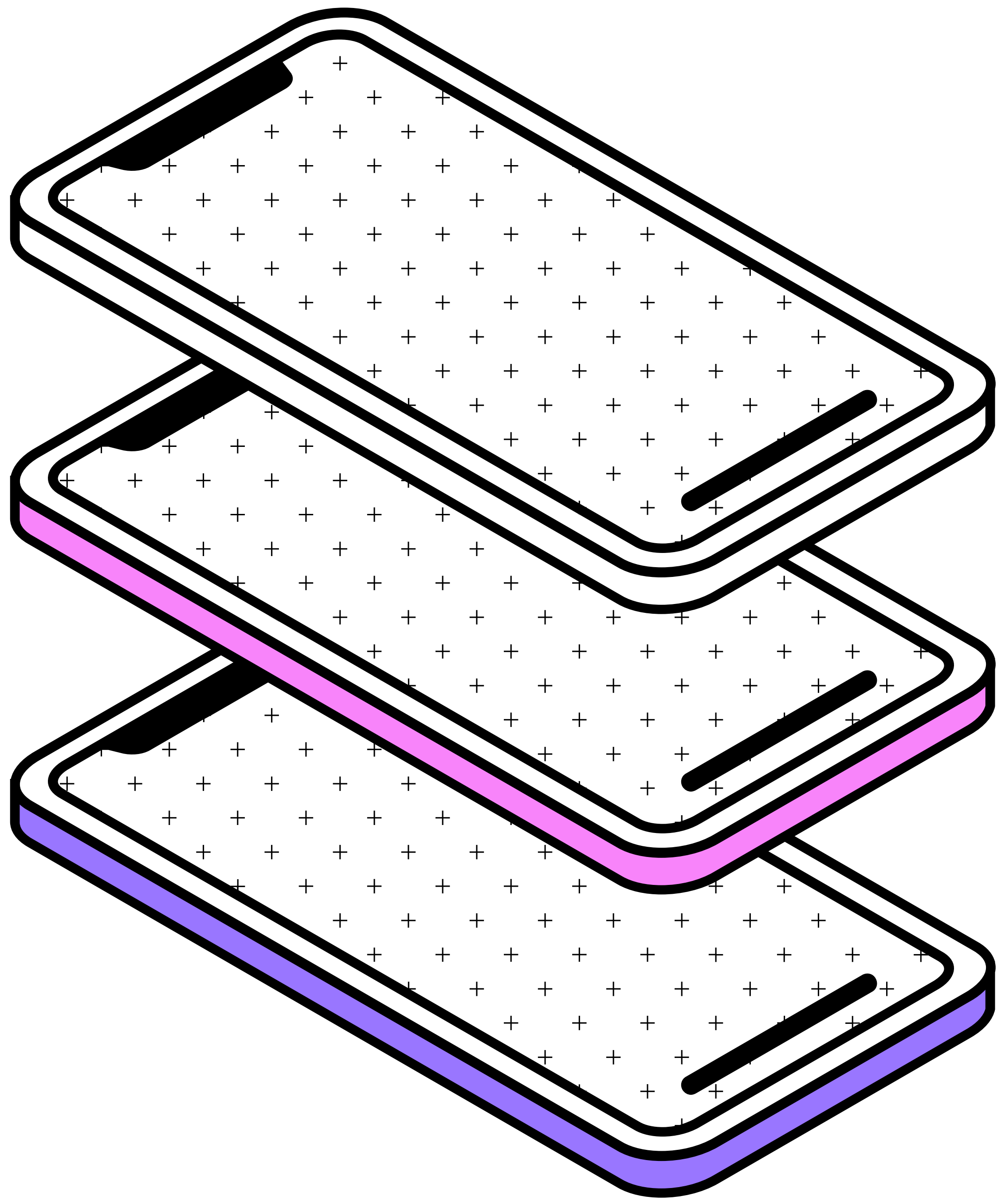
Parameter Control in Evolutionary Algorithms

Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework

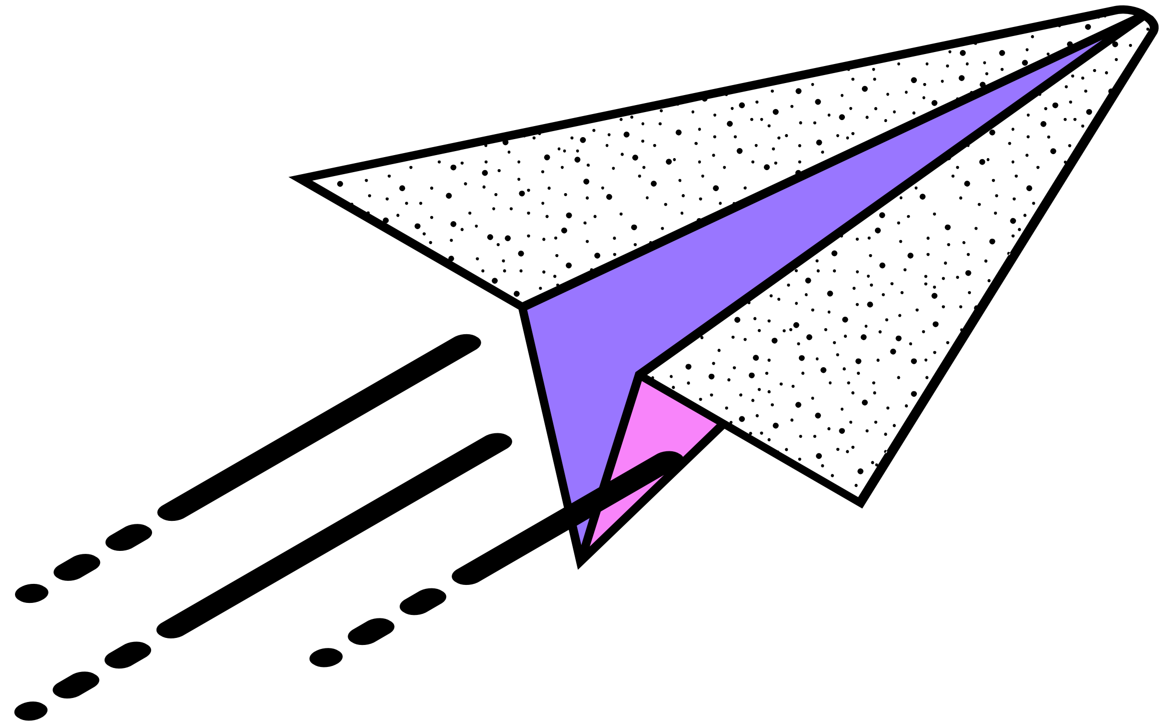
Lecturer: PhD. Lương Ngọc Hoàng

Group 9:

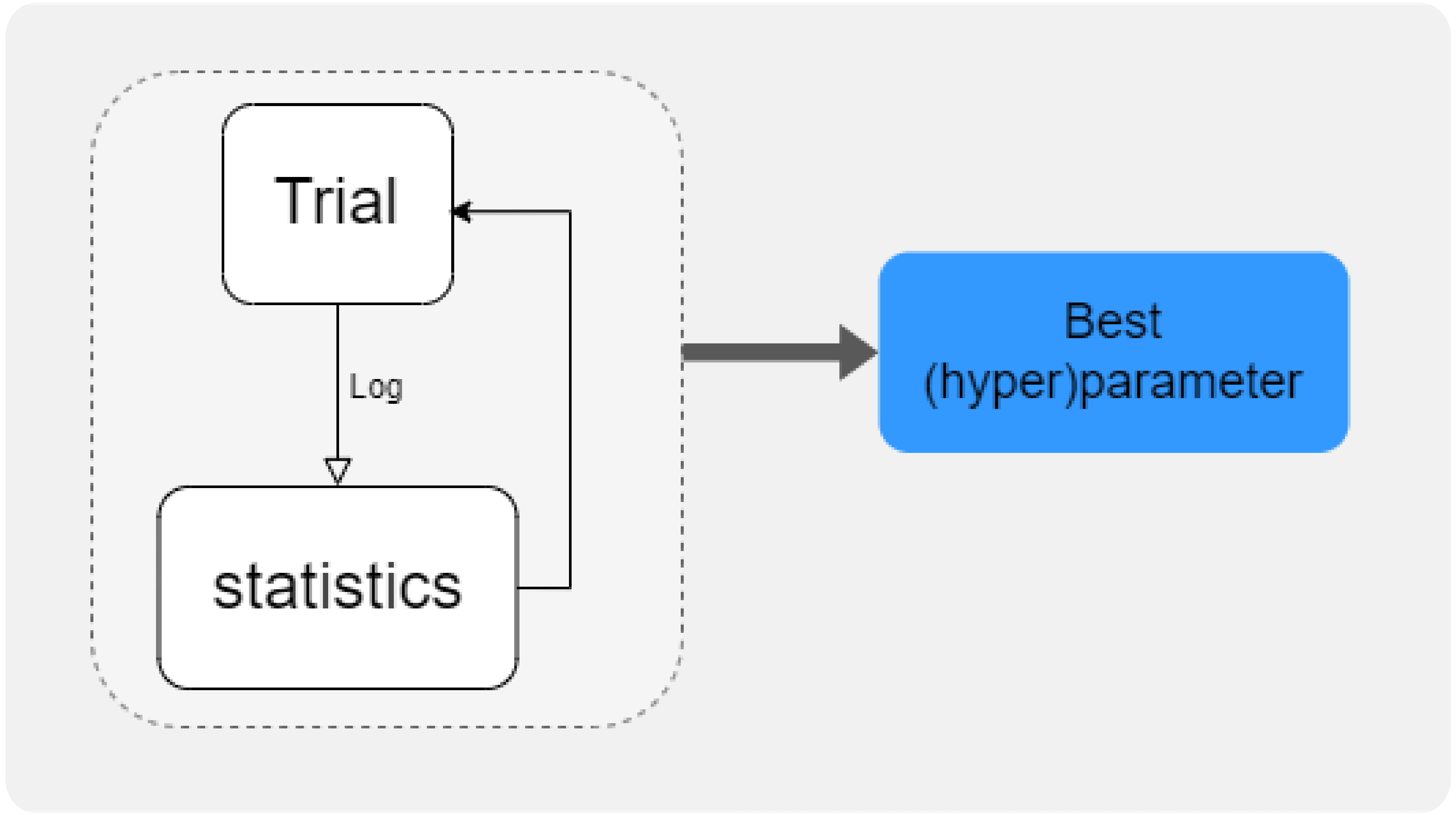
Đặng Quang Hưng - 230101006
Lê Trường Long Hưng - 230101007
Võ Ngô Văn Tiền - 230102033
Trần Văn Tịnh - 220101039



Today's Agenda



I. Introduction



Static configurations:

- The configuration will only work well when the configured algorithm will be used to solve similar tasks.
- The iterative nature of most algorithms is ignored and a configuration is assumed to work best at each iteration of the algorithm

0 Gold

0 Silver

0 Bronze

Algorithm **Parameter** **Instances** **Objective**

1 Gold

0 Silver

0 Bronze

Algorithm **Parameter** **Instances** **Objective**

Algorithm

Parameter

Instances

1 Gold

1 Silver

0 Bronze

Objective

1 Gold

0 Silver

0 Bronze

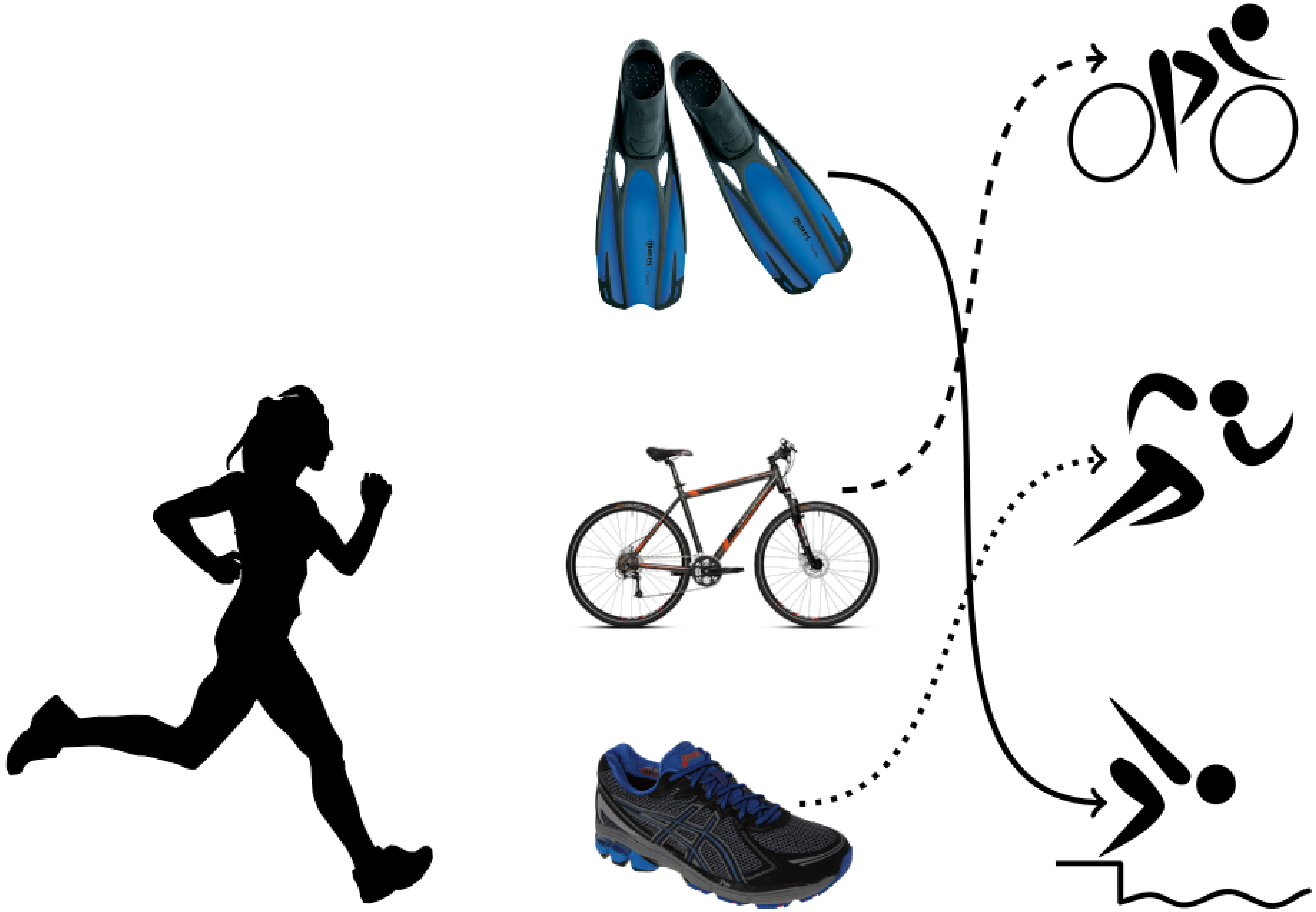
Algorithm **Parameter** **Instances** **Objective**

1 Gold

1 Silver

0 Bronze

Algorithm **Parameter** **Instances** **Objective**

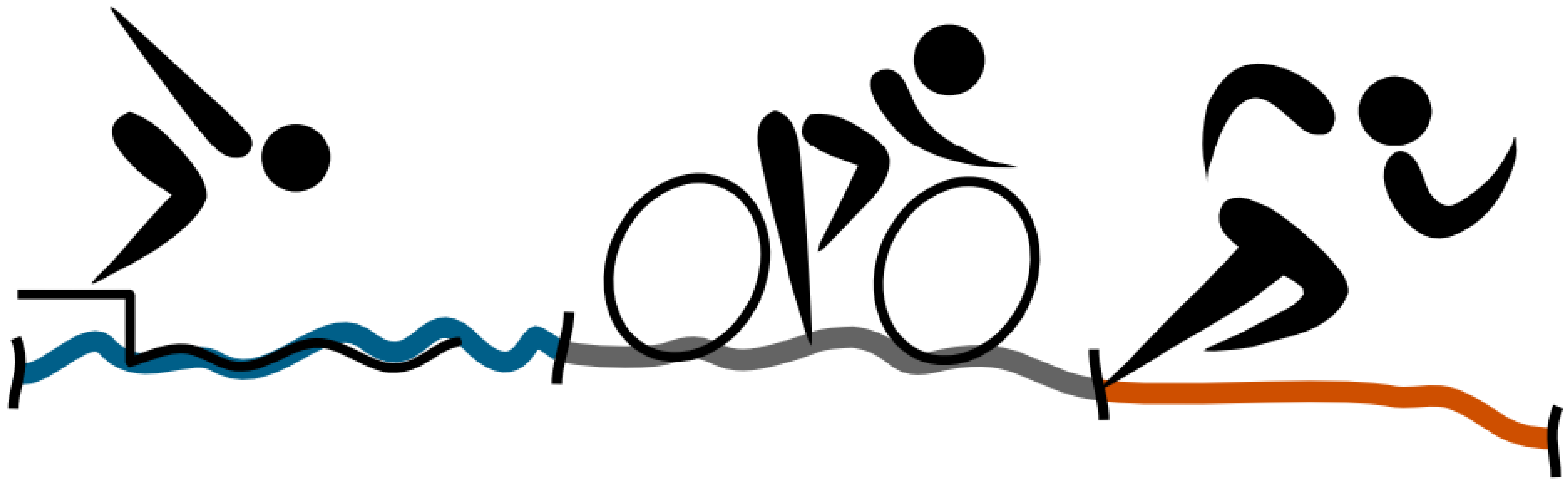


3 Gold

0 Silver

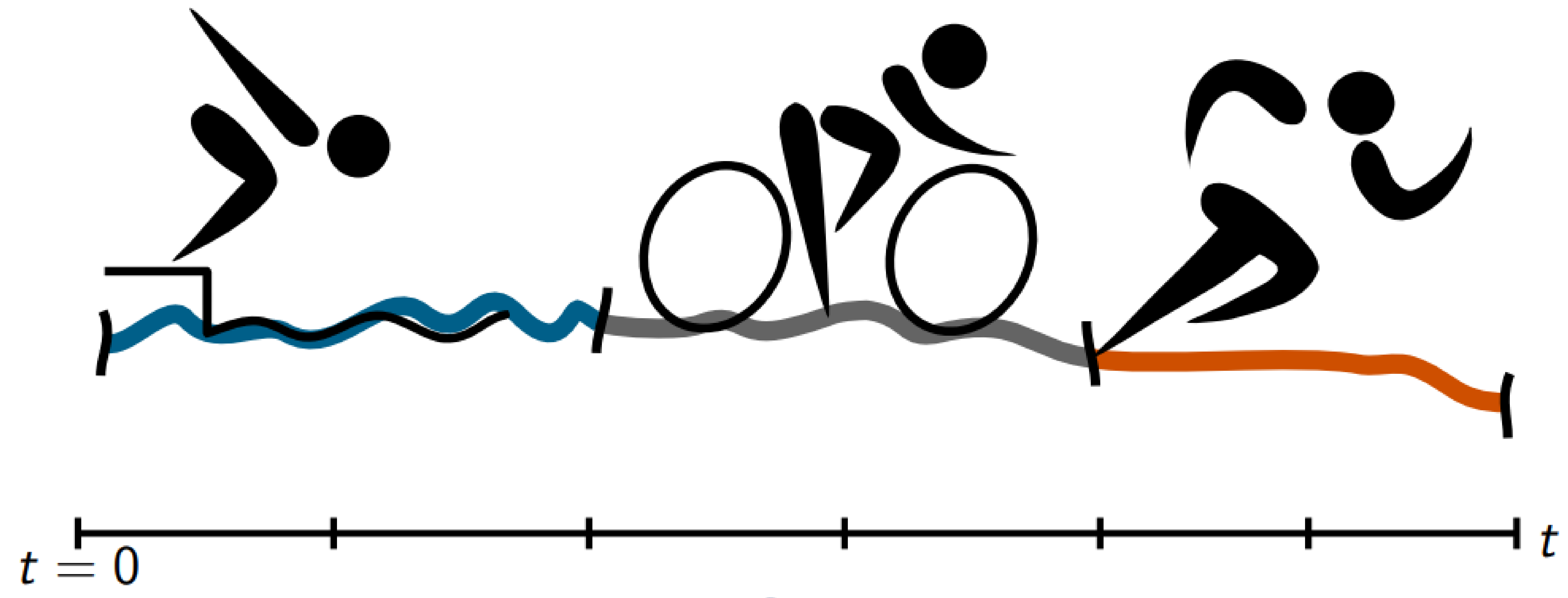
0 Bronze

Algorithm **Parameter** **Instances** **Objective**

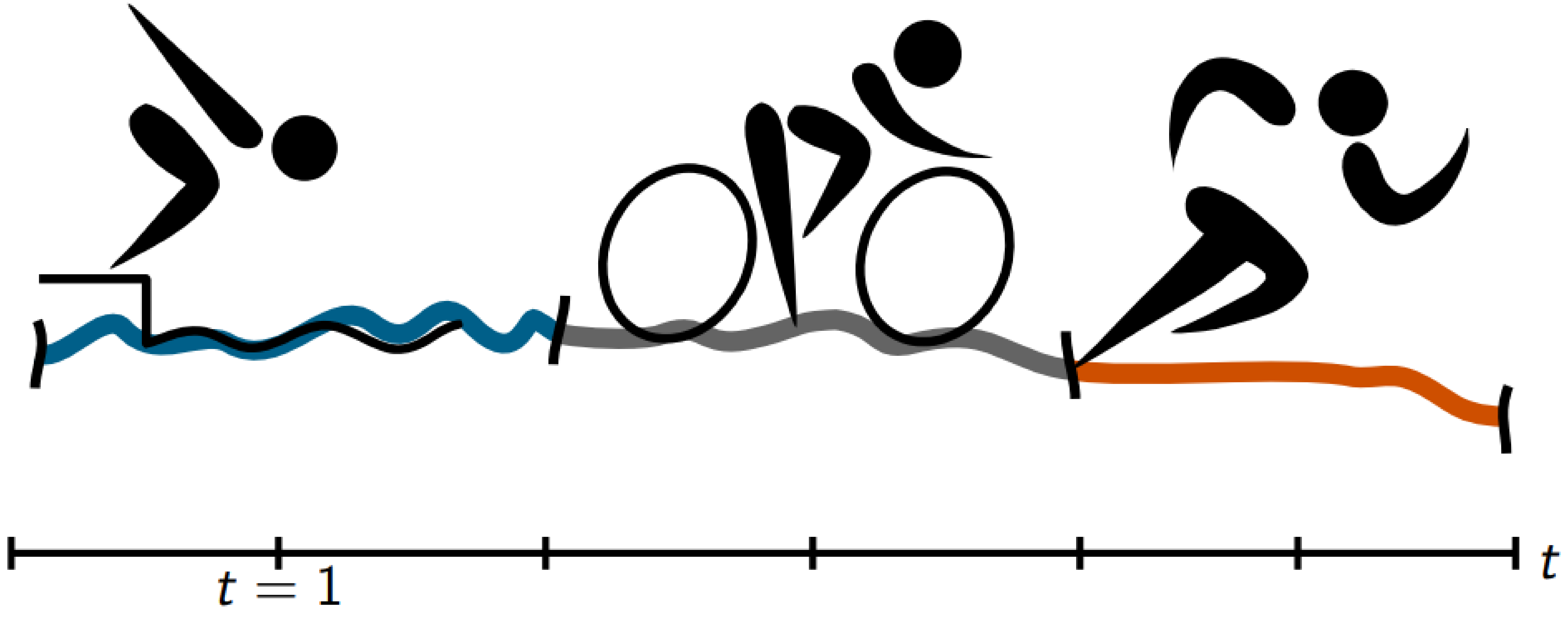


How can we solve such problem instances?

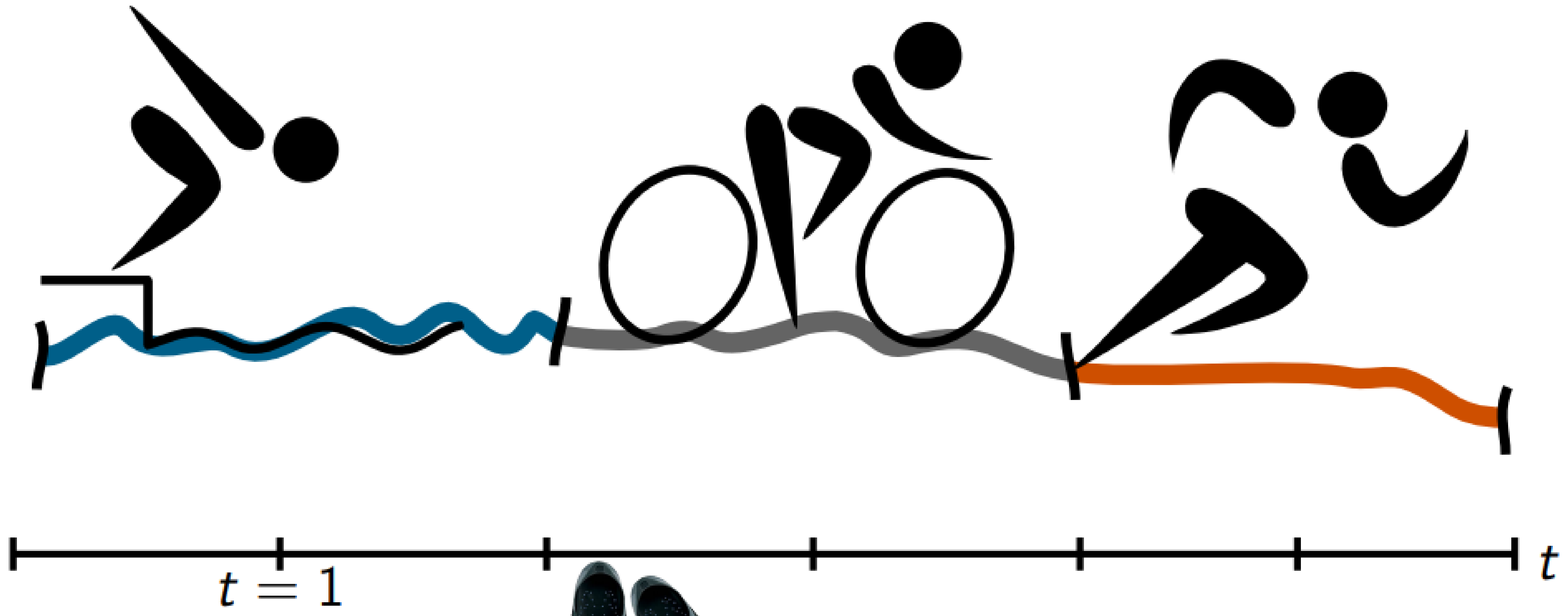
Dynamic Algorithm Configuration (DAC)



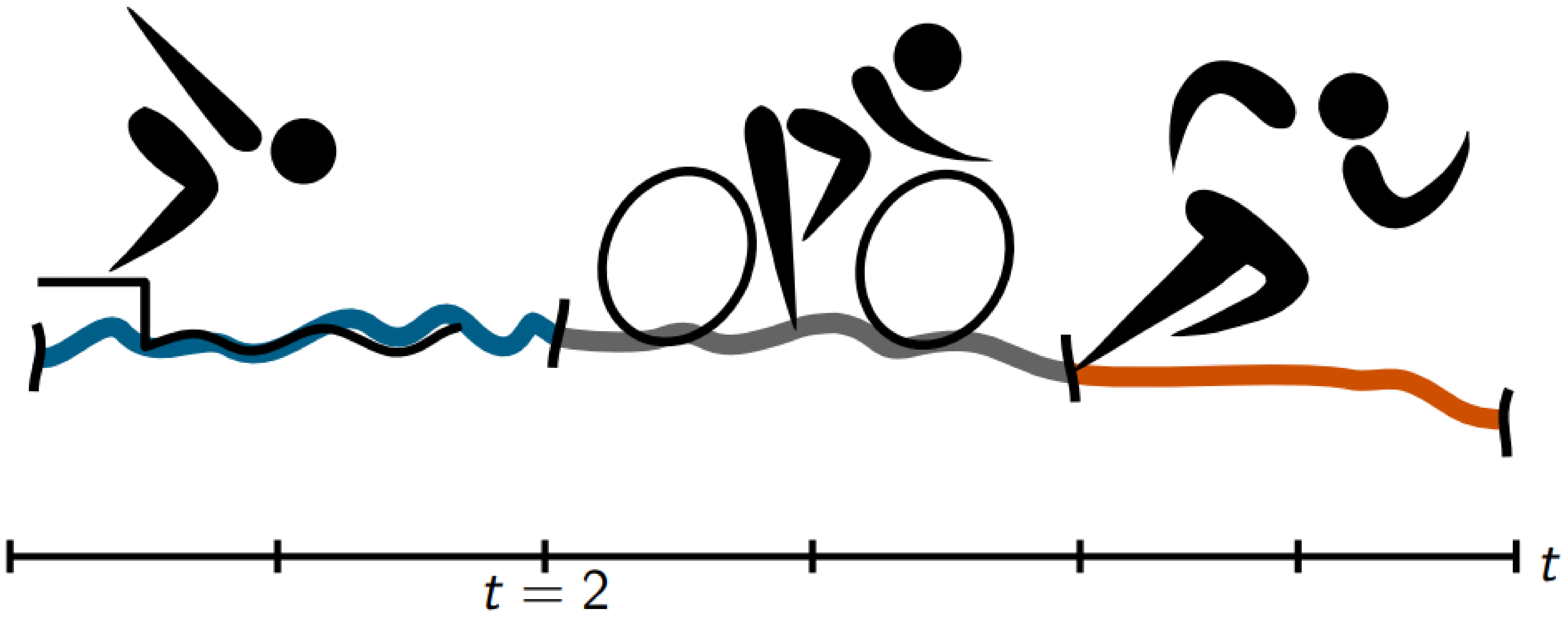
Start from some default:



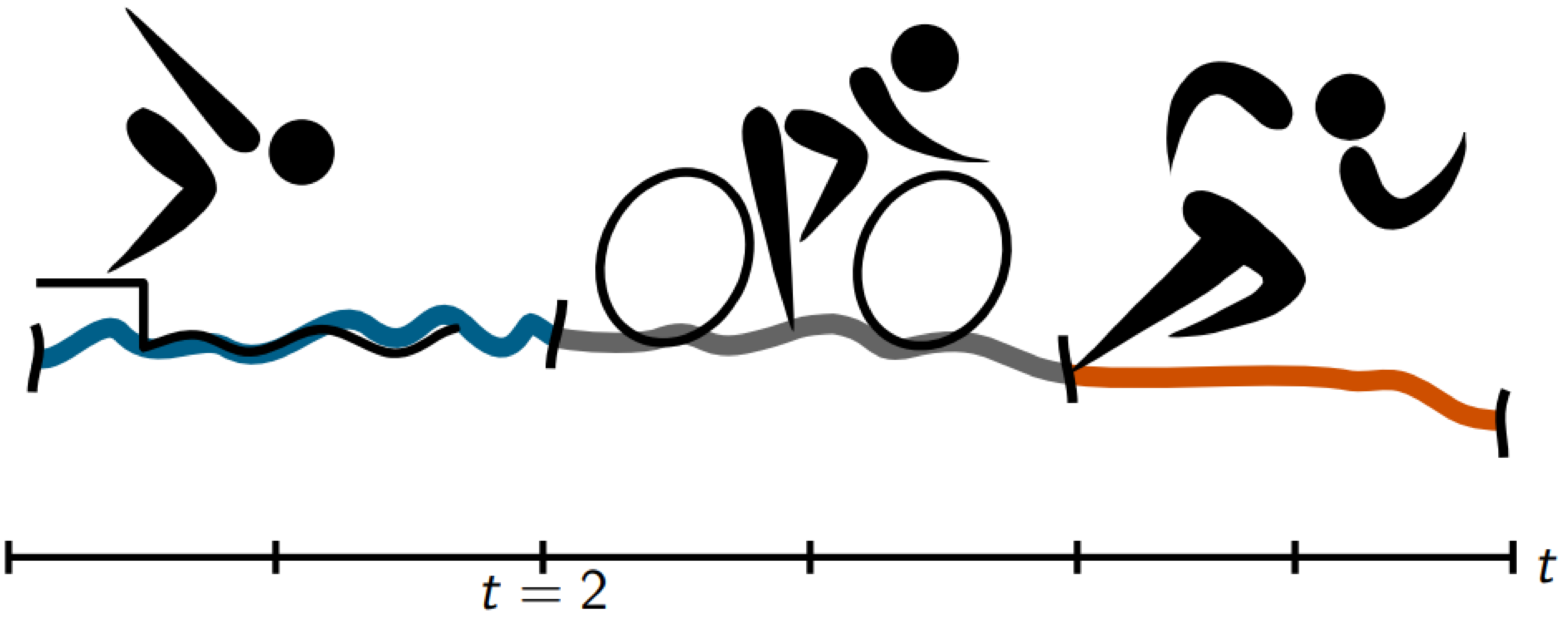
Observe state: Water



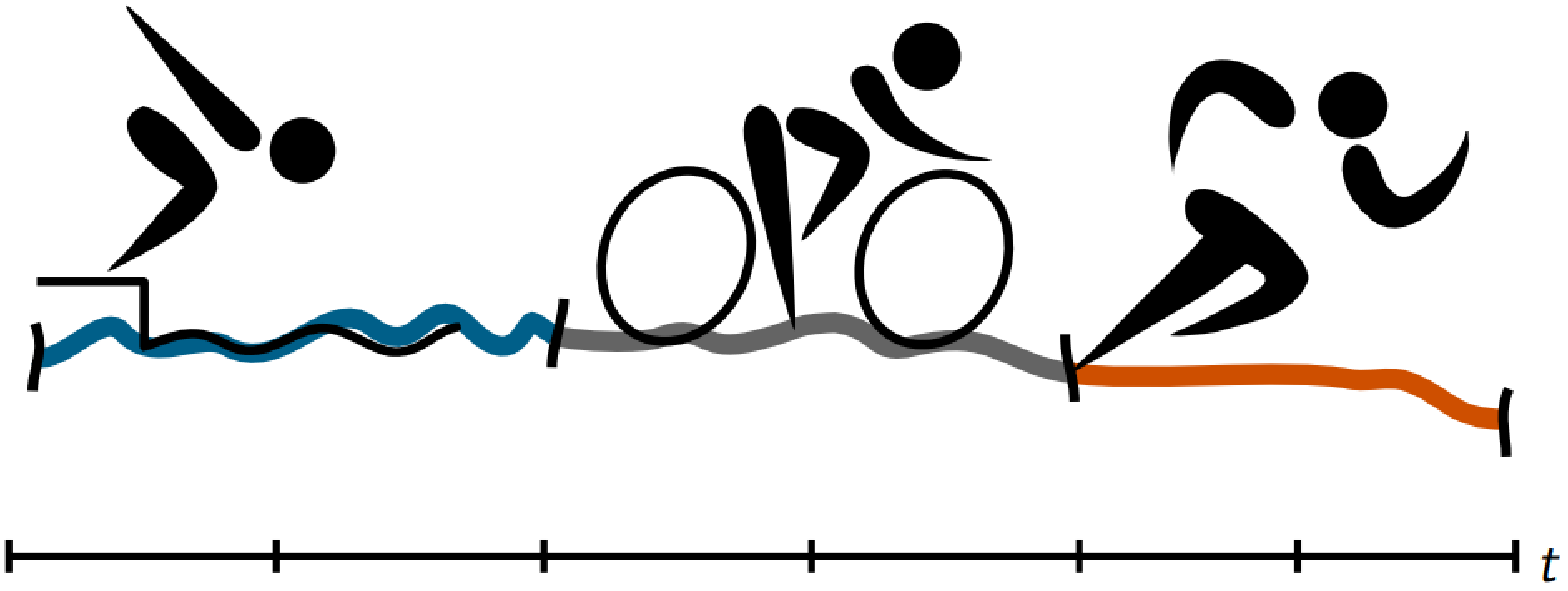
Adjust parameter:



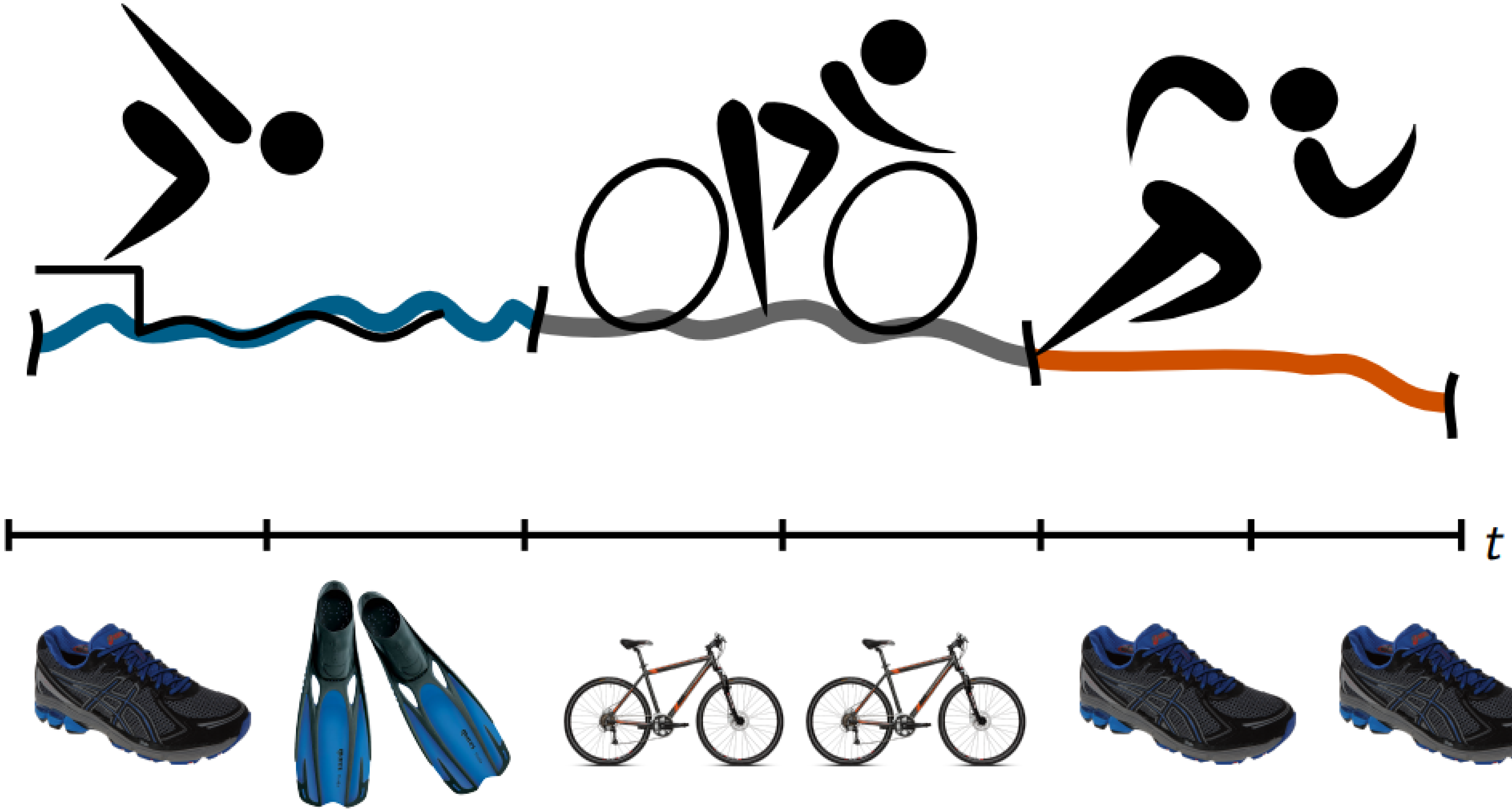
Observe state: Bike Trail



Adjust parameter:


t

...


t

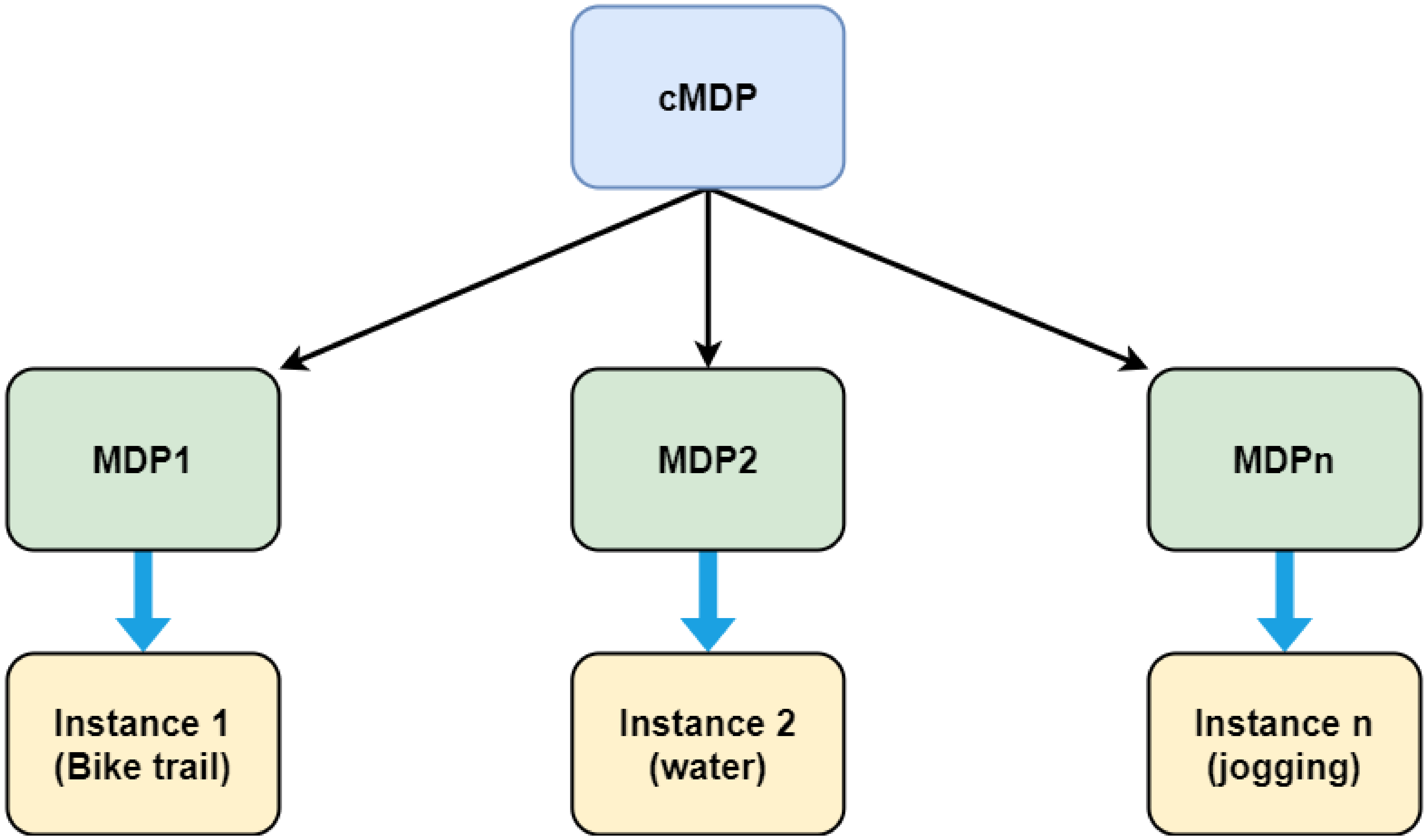
II. DAC as Contextual MDP

DAC is modeled as Contextual Markov Decision Process (CMDP)

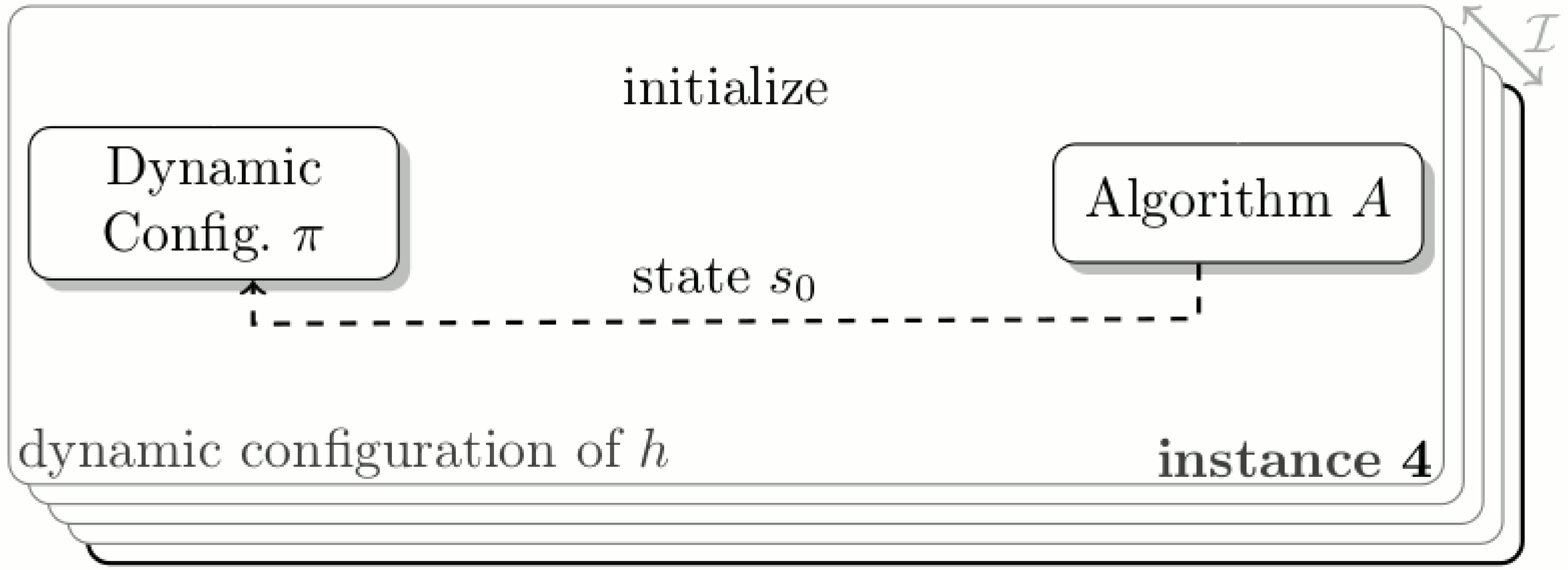
MDP **M** is a tuple ⟨**S, A, T, R**⟩:

- **S** (states): a set of states that the system can transition between.
- **A** (Actions): a set of actions that can be executed when the system is in a specific state.
- **T** (Transition Function): defines the probability of transitioning from one state to another when an action is executed
- **R** (Reward Function): determines the reward or benefit obtained when an action is taken in a specific state.

II. DAC as Contextual MDP



II. DAC as Contextual MDP



II. DAC as Contextual MDP

$$\begin{aligned}\mathcal{V}_i^{\pi}(s_t) &= \mathbb{E}\left[r_{t+1}(i) + \gamma \mathcal{V}_i^{\pi}(s_{t+1}) | s_{t+1} \sim \mathcal{T}_i(s_t, \pi(s_t))\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}(i) | s_t = s\right]\end{aligned}$$

Challenges in DAC

- Budget constraints for running an algorithm until a cutoff is reached.
- Varying lengths of algorithm runs depending on the effectiveness of the chosen parameter settings.
- Parameter interaction effects where the choice of one parameter value influences others
- Varying degrees of homogeneity of the instances.
- Noisy rewards.

III. White-Box Benchmarks for DAC

Sigmoid

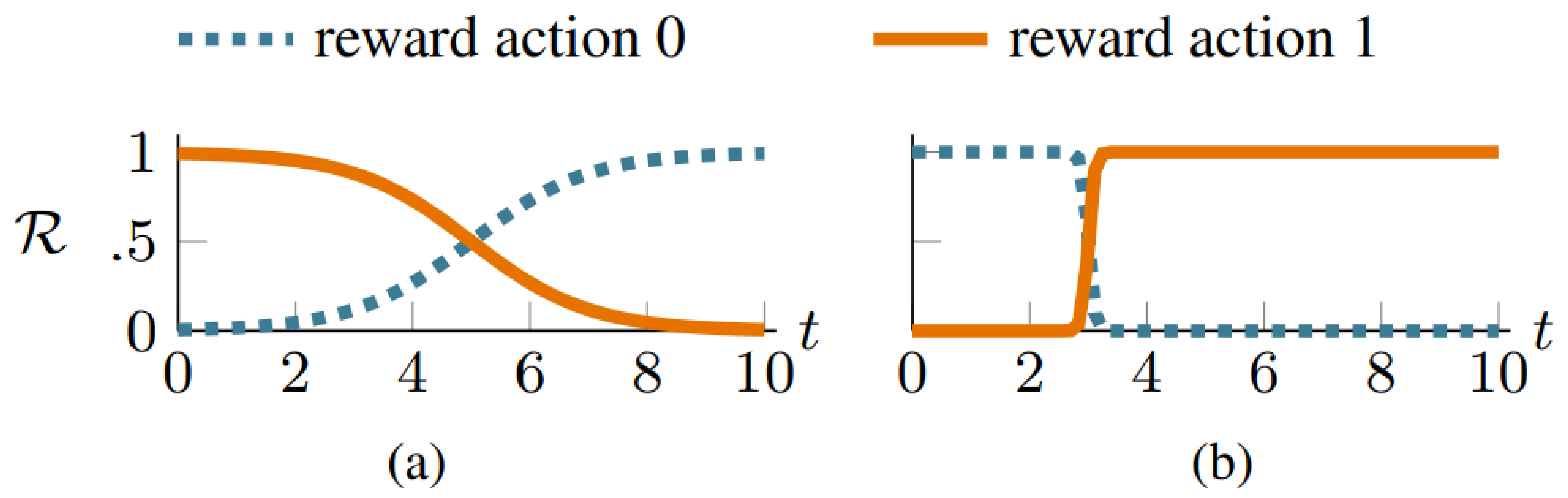
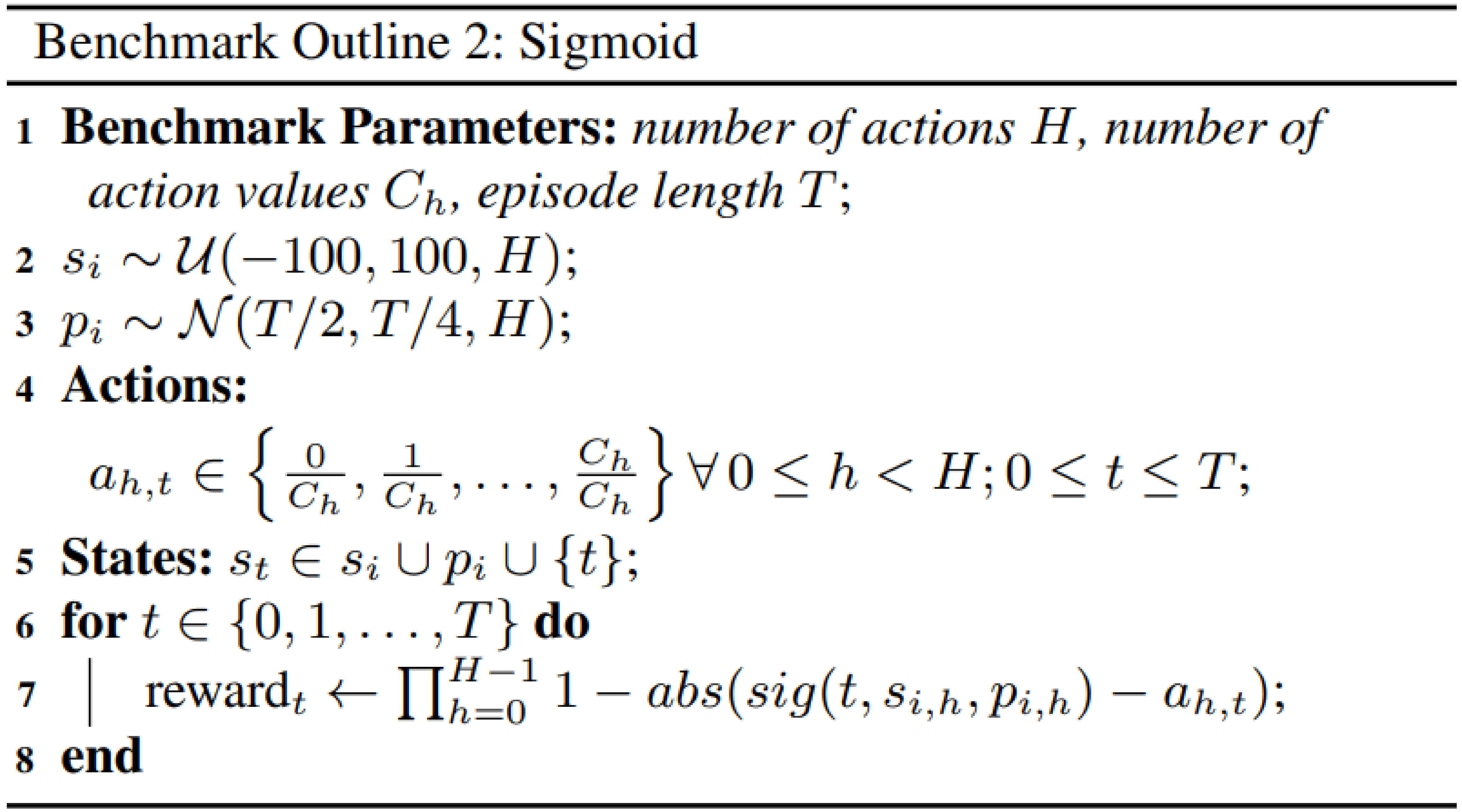


Figure 1: Example rewards for Benchmark *Sigmoid* with $T = 10$, $N = 1$ and $a_{0,t} \in \{0, 1\}$ on two instances, where $p_{(a)} = 5$, $s_{(a)} = 1$ on (a) and $p_{(b)} = 3$, $s_{(b)} = -20$ on (b). The solid/dashed line depicts the reward for action 1/0 at time-step t. On (a) it is preferable to select action 1 for the first halve of the sequence whereas on (b) it is better to start with action 0.

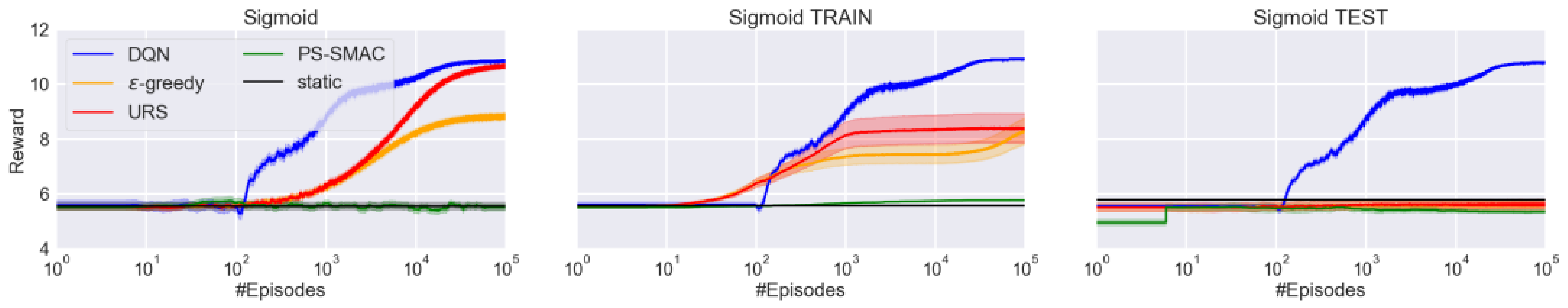
Sigmoid



Benchmark Outline 2: Sigmoid

```
**Benchmark Parameters:** *number of actions* $H$, *number of action values* $C_h$, *episode length* $T$;
$s_i \sim \mathcal{U}(-100, 100, H)$;
$p_i \sim \mathcal{N}(T/2, T/4, H)$;
**Actions:**
    $a_{h,t} \in \left\{\frac{0}{C_h}, \frac{1}{C_h}, \ldots, \frac{C_h}{C_h}\right\} \forall 0 \leq h < H; 0 \leq t \leq T$;
**States:** $s_t \in s_i \cup p_i \cup \{t\}$;
**for** $t \in \{0, 1, \ldots, T\}$ **do**
    reward$_t \leftarrow \prod_{h=0}^{H-1} 1 - abs(sig(t, s_{i,h}, p_{i,h}) - a_{h,t})$;
**end**
```

Sigmoid



(a) Sampling from distribution (b) Fixed Training Set (c) Unseen Test Instances

Figure 2: Comparison of generalization to new instances on 1D-*Sigmoid* with binary action space and $T = 11$. The solid line represents the mean reward and the shaded area the standard error over 25 repetitions. To estimate the performance over the distribution of instances in (a), we sample 10 new random sigmoid functions for evaluation. In (b) we evaluate the agents on 100 training instances. In (c) we show the generalization capability by evaluating the agents on 100 new, prior unseen test instances, evaluating them after every training-step in (b) without additional training. A random policy could only achieve a reward of 5.5. The performance of the optimal static policy is given in black.

IV. Discussion

- **Feasibility of RL for DAC:**
    - RL can be computationally expensive and may not be suitable for all scenarios.
- **DAC requires informative state features**
    - It is important to have informative state features based on which a policy can change parameter configurations.
- **Reward design**
    - Reward design is crucial for DAC, but quality approximation can be difficult in some domains.

V. Conclusion

- Propose a general framework which can learn configuration policies across instances
- Introduce new white-box benchmarks, which enable to study DAC with a variety of different properties.
-
- Robustness of Reinforcement Learning (RL) for DAC.

VI. References

[1] André Biedenkapp, H. Furkan Bozkurt, Theresa Eimer, Frank Hutter, Marius Lindauer: Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework. ECAI 2020: 427-434

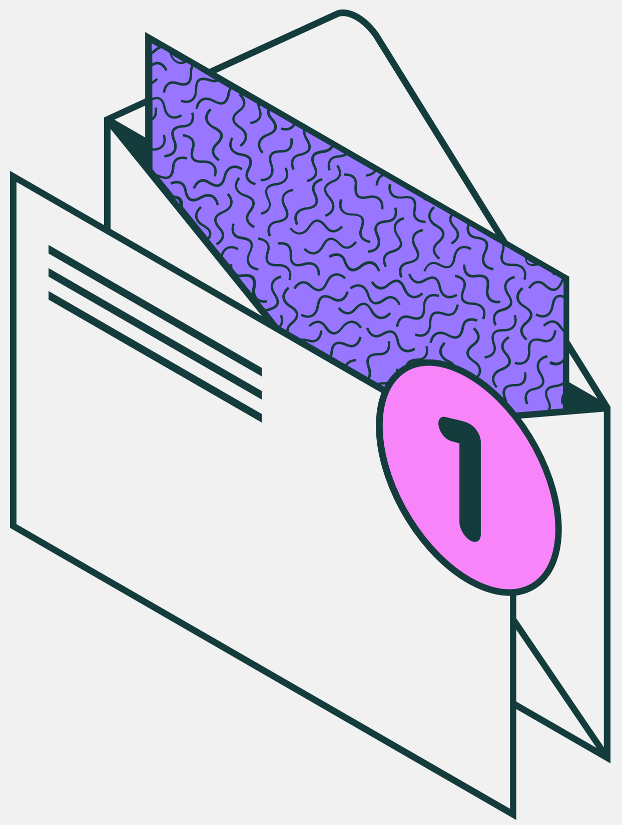
[2] https://github.com/automl/DAC/blob/master/Appendix.pdf

[3] https://www.automl.org/dynamic-algorithm-configuration/

[4] https://www.automl.org/automated-algorithm-design/dac/

[5] André Biedenkapp, https://andrebiedenkapp.github.io/assets/pdf/slides/2020_DAC@ECAI.pdf

Thanks for listening



Do you have any questions?