Heterogeneous Sensor Placement Under Uncertainty

AMIN JABINI and ERIK A. JOHNSON

ABSTRACT

This study presents a heterogeneous sensor placement optimization framework using deep reinforcement learning (DRL) that considers system parameter uncertainty. The sensor placement problem is a well-established combinatorial optimization problem characterized by inherent parameter uncertainties that affect system responses that sensors measure. These uncertainties render deterministic solutions insufficient and necessitate a computationally tractable approach to account for the uncertainties. The proposed method incorporates a Markov decision process (MDP) as a stochastic environment, and a sensor placement agent trained using DRL. The agent's objective is to maximize the effectiveness of sensor placement within a system by selecting sensor types and locations. The agent's sequential decision-making is guided by a reward function that is designed based on the observability Gramian, calculated using sampled parameter values from an a priori distribution. The proposed approach is validated through simulation of a case study involving heterogeneous sensors in a shear building model with results compared to those from an evolutionary algorithm. The results show that the sensors selected by DRL method match the CMA-ES algorithm with the advantage of having information about relative importance of the selected sensors.

INTRODUCTION

Sensor networks are critical components of engineering systems, providing valuable data for informed decision-making. The types of sensors used in various engineering applications are diverse, including temperature, pressure, motion, and chemical concentration sensors. Collected data can be used for diverse purposes, such as fault detection, environmental monitoring, and process optimization. The effective design of sensor networks is crucial to making informed decisions based on sensor data. In practice, the number of sensors is often limited by budget constraints, so it is essential to carefully choose the locations of sensors to obtain the most valuable information. This objec-

Amin Jabini, PhD Student, Sonny Astani Department of Civil and Environmental Engineering, University of Southern California, Los Angeles, CA, USA. Email: jabini@usc.edu. Erik A Johnson, Professor, Sonny Astani Department of Civil and Environmental Engineering, University of Southern California, Los Angeles, CA, USA.

tive forms an optimization problem, where the goal is to find the most cost-effective sensor configuration, i.e., the most informative set of measurements [1]. The discrete optimization problem of sensor placement is NP-hard [2]. The number of solutions scales factorially in the number of sensor location-type pairs. For example, for a system with 20 possible sensor locations, the number of choices for 6 sensors of 3 sensor types exceeds 50 million. This problem becomes even more computationally costly when robust decisions under uncertainties are crucial, such as robot sensor placement where considering worst-case scenarios in sensor decision-making is important [3]. The objective of the optimization problem in sensor placement depends on the application and the goal of the instrumentation. In structural health monitoring, numerous criteria have been utilized [4]. Some researchers have used measures based on the Fisher Information Matrix (FIM). Since FIM is related to the expected estimation covariance by the Cramer-Rao inequality, objectives such as the minimum eigenvalue, the determinant or the trace of the inverse of FIM are connected to the lower bound for the estimation uncertainty. Other researchers used objectives based on the observability Gramian [5]. The minimum eigenvalue of the observability Gramian is a measure of the output energy of the least observable state [6]. Multiple optimization methods have been proposed in the literature. An early approach is to use forward or backward sequential optimization algorithms [7], where sensors are added or removed at each step, observing the behavior of the cost function. However, the nonlinear and dependent nature of the interaction between sensor locations often results in suboptimal solutions, and the behavior of the cost function may not be monotonic for all cost functions. Another category of methods commonly used is mixed integer programming, which has been shown to scale exponentially in the number of binary variables and is not computationally tractable for large systems [6]. A popular category of approaches is meta-heuristic algorithms, such as swarm intelligence-based methods [8]. However, these methods require significant computing power and cannot guarantee a globally optimal solution. Another disadvantage is that the results of these methods are not reusable in similar systems.

The recent breakthroughs in deep reinforcement learning (DRL) in decision making and optimization problems have attracted the interest of researchers across various engineering domains. DRL has demonstrated promising performance in optimization problems, including resource management [9], chip design [10], and life-cycle maintenance (Andriotis and Papakonstantinou 2019). Herein, we formulate the fixed-budget heterogeneous sensor placement problem under uncertainty with a Markov decision process (MDP), where the state is a binary vector of sensor configurations, and actions are the integers of sensor type-location pairs. We apply a DRL algorithm called deep Q-Network (DQN) to train the sensor selector agent. We consider the normalized observation equation and use the increase of the minimum eigenvalue of the normalized observability Gramian by each action as the reward for each step. We simulate this approach for a small-scale 3-DOF system with three sensor types and compare the results with a genetic algorithm using CMA-ES strategy [11] that is known to work with stochastic fitness functions.

PROBLEM FORMULATION

For a system with uncertain parameters $\boldsymbol{\theta}$, n candidate sensor location, and p available sensor types, the objective is to select m sensor location-type pairs. The sensor configuration can be represented by a binary state vector s, which is part of a finite state set $\mathcal{S}$. In a greedy approach, sensors are added sequentially within a horizon of m. At each step, a location-type pair is selected from an action set $\mathcal{A}$, i.e., the integers from 1 to np, and the corresponding element of the state vector is updated to have a value of 1. In a non-greedy approach, such as CMA-ES, the population consist of binary vectors, and the budget constraint is incorporated into the fitness function by with a large penalty function.

Deep Reinforcement Learning

To solve the heterogeneous sensor placement problem with DRL, the Markov decision process (MDP), which uses transition probability model P based on the probability of sensor failure (which is assumed to be 0 in this paper) and a reward function $r\left(\mathrm{s}, a\right)$. For a sensor-location pair represented by action a, the next state s' is $\mathrm{s} + \mathbf{e}_{\mathrm{a}}$ with the probability of $1 - p_f\left(a\right)$ or s with the probability $p_f\left(a\right)$, where $p_f\left(a\right)$ is the probability of failure of the sensor selected in action a and $\mathbf{e}_a$ is a unit vector with 1 in index corresponding to action a. In this setting, the agent selects an action at each step, the environment updates the state, and a reward denoted by $r\left(\mathrm{s}, a\right)$ is provided to the agent. This process is repeated for m steps in each episode. The agent stores the collected transition data, i.e., tuples of $\left(\mathrm{s}_t, a_t, \mathrm{s}_{t+1}\right)$ in its buffer and uses it to update its policy. The objective of the agent is to maximize its expected future reward. The agent's policy can be represented by the function $\pi_\phi\left(a_t|\mathrm{s}_t\right)$, which is parameterized by vector ϕ. The agent's goal can be expressed as

$$J\left(\phi\right) = E_{\tau\sim p(\tau)}\left[\sum_{t=1}^{m}\gamma^{t-1}r\left(\mathrm{s}_t, a_t\right)\right] \tag{1}$$

in which γ is the future discount factor, τ is the trajectory $[a_1, \mathrm{s}_1, r_1, \ldots, a_m, \mathrm{s}_m, r_m]$. A category of reinforcement learning methods is based on a value function approximation. Let $Q\left(\mathrm{s}_t, a_t; \phi\right)$ be the Q-function, which estimates the true Q-values $Q\left(\mathrm{s}_t, a_t\right)$ that are defined as expected value of action a_t in state s_t, and let ϕ be the weights of the neural network that represents the Q-function. In standard Q-learning, the update rule for the Q-function is given by:

$$Q\left(\mathrm{s}_t, a_t; \phi\right) \leftarrow Q\left(\mathrm{s}_t, a_t; \phi\right) + \alpha\left[r\left(\mathrm{s}_t, a_t\right) + \gamma\max_{a_{t+1}}Q\left(\mathrm{s}_{t+1}, a_{t+1}; \phi\right) - Q\left(\mathrm{s}_t, a_t; \phi\right)\right] \tag{2}$$

where α is the learning rate. In the DQN algorithm, two separate Q-functions are used: target network $Q\left(\mathrm{s}_t, a_t; \phi\right)$, and prediction network $Q'\left(\mathrm{s}_t, a_t; \phi'\right)$. At each time step, the target network is used to select the action with the highest Q-value (in practice, using an ϵ-greedy approach), and the prediction network is used to estimate the value of that action in state $\mathbf{s}_t$. The update rule of the prediction network is as follows:

$$Q'\left(\mathbf{s}_t, a_t\right) \leftarrow Q'\left(\mathbf{s}_t, a_t\right)+\alpha\left[r\left(\mathbf{s}_t, a_t\right)+\gamma Q'\left(\mathbf{s}', \operatorname{argmax}_{a_{t+1}} Q\left(\mathbf{s}_{t+1}, a_{t+1}\right)\right)-Q'\left(\mathbf{s}_t, a_t\right)\right] \tag{3}$$

in which the parameters ϕ and ϕ' are dropped from the corresponding Q-functions for notational simplicity. The target network is updated with a fixed frequency by updating the parameters ϕ to be ϕ'. The reward function is based on the minimum eigenvalue of the observability Gramian, denoted by $\lambda_{\text{min}}(\mathbf{W}_{\text{o}})$, for the normalized observation equation $\widetilde{\mathbf{y}} = \mathbf{R}^{-1/2}\mathbf{y}_t = \mathbf{R}^{-1/2}\,\mathbf{H}(\mathbf{s}_t;\theta)\mathbf{x}+\mathbf{R}^{-1/2}\nu$, where $\mathbf{R}$ is the covariance of the zero mean Gaussian noise ν. This normalization makes the observability equation and, hence, the observability Gramian dimensionless. Therefore, different measurement types become comparable. This also incorporates the sensor quality in the reward function. In each episode, the values of uncertain parameters are sampled from associated probability distributions and the reward function is calculated based on the sampled values. In order to make the reward non-sparse, instead of the lumped-sum reward in the final step of selecting the sensors, the reward function is defined sequentially as:

$$r(\mathbf{s}_t, a_t;\theta) = \widetilde{\lambda}_{\text{min}}(\mathbf{s}_{t+1};\theta) - \widetilde{\lambda}_{\text{min}}(\mathbf{s}_t;\theta) \tag{4}$$

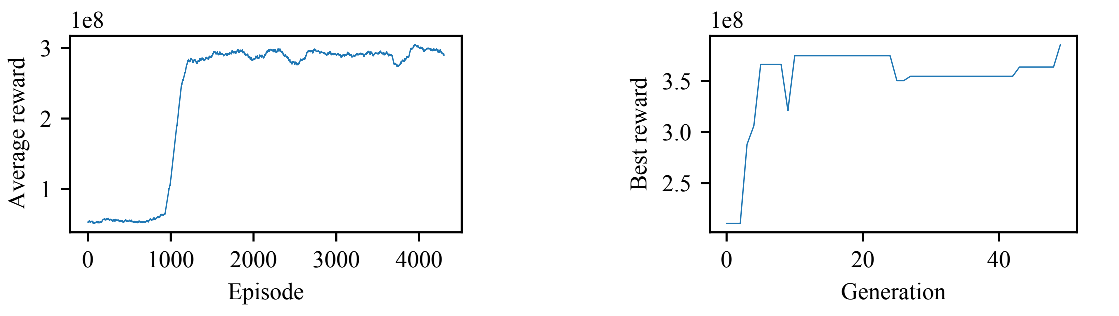
in which $\widetilde{\lambda}_{\text{min}}(\mathbf{s}_{t+1};\theta)$ is the minimum eigenvalue of the normalized observability Gramian calculated based on sample values θ.

Covariance Matrix Adaptation Evolution Strategy

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11] is used as a gradient-free optimization method to find the best solution in a high-dimensional space. CMA-ES generates individuals with a probabilistic search strategy, updates the mean and the covariance matrix based on the quality of evaluations [12], and explores the search space to converge to the optimal solution for stochastic objective functions. The method is adapted to binary vectors by projecting search points to the binary vector space. The fitness function is the minimum eigenvalue of the normalized observation. Solutions with more than m sensors are given a large negative fitness.

RESULTS

The DRL framework for sensor placement is applied to a 3-DOF shear building structure that is intended to be equipped with three sensors of three possible sensor types: drift, drift velocity, and absolute acceleration. The diagonal values of noise covariance for each of these sensor types are 3×10^{-7} m^2, 10^{-5} m^2/s^2, and 3×10^{-3} m^2/s^4 respectively. The agent's policy is constructed using a three-layer neural network with 12 units in the hidden layer and 9 units in the input and output layers, that corresponds to 9 sensor type-location pairs. The activation functions are Rectified Linear Unit (ReLU) and linear function for the first two layers, respectively. The value of each unit in the output layer represents the corresponding Q-value for each action. The agent is trained for 4500 episodes. Figure 1a shows the average performance of the agent in training, plotted as a moving average with a window size of 200, and Figure 3 illustrates the exploration



(a) Average reward of DQN algorithm (b) Best individual reward in CMA-ES

Figure 1. Comparison of the performance of DRL and CMA-ES methods

in sensor configuration space. Both of these figures show that the agent has converged after 1500 episodes and the occasional exploration is due to ϵ-greedy action selection. The sensors ultimately selected by the trained policy are drift velocity sensors for the second, first, and third floors in sequential order. This includes the sequential order since the discount factor γ is less than one and reward function is designed to be sequential. The agent's performance demonstrates parity with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm, as depicted in Figure 1b. The CMA-ES algorithm was simulated with a population size of 80, selecting the top 40 individuals in each generation to produce offspring. The algorithm was configured with a crossover probability and mutation rate set to 0.6 and 0.15 respectively. The initial search points were generated based on a Normal distribution, with a mean of 0.5 and a standard deviation of 1 for each vector element. Figure 2 provides a visual representation of the average state of the top 40 individuals across the simulation. The selected sensors in CMA-ES validates those selected by DRL but, unlike DRL, it doesn't have any information about their sequential importance.

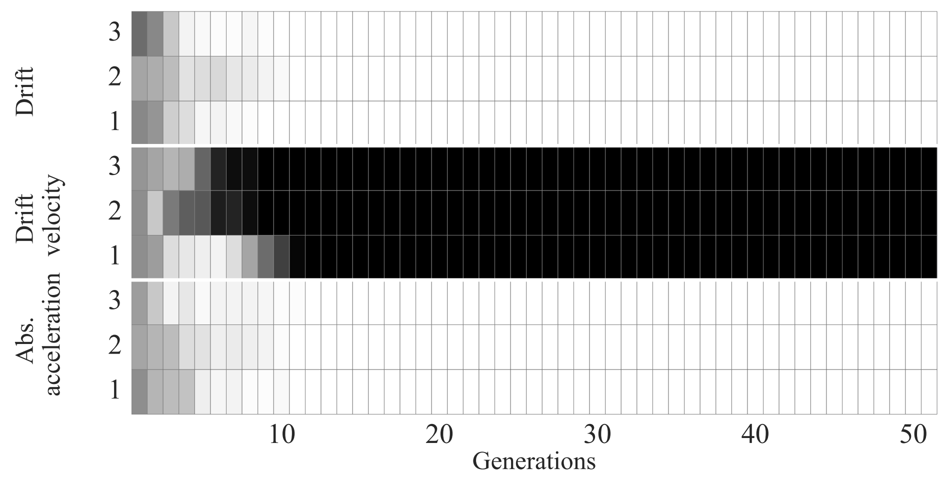


Figure 2. The average of top 40 individuals in generations

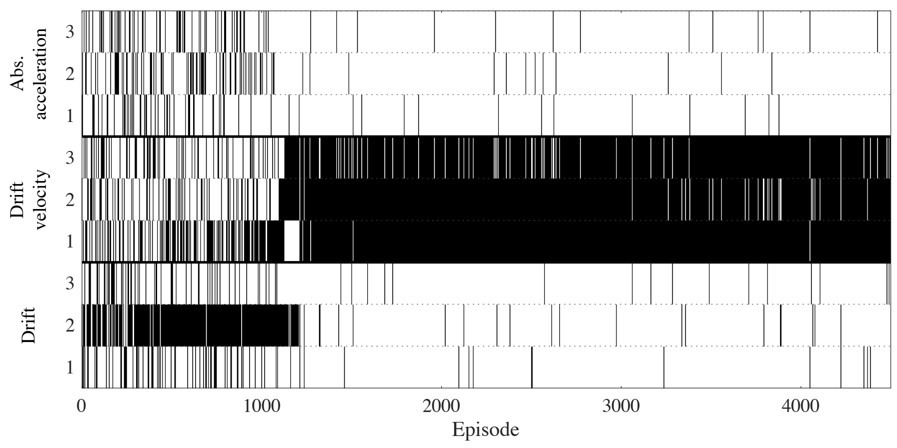


Figure 3. The explored sensor configurations during the training

CONCLUDING REMARKS

This study uses DRL as a stochastic optimization framework for a heterogeneous sensor placement problem utilizing normalized observability Gramian enables comparing different sensor types. The framework is applied to a synthetic example of 3-DOF structure and the results are validated with an evolution-based method. DRL has the advantage of reusablity of the output, i.e., the trained policy, may be applied to similar problems through transfer learning. Another advantage is that use of a function approximation in DRL results in updating the approximated values for unseen states by updating the shared function parameters based on observed states. In addition, the exploration and exploitation is easily configurable through an ϵ-greedy approach.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support of this work by National Science Foundation (NSF) through grant 16-63667, and the first author's Annenberg Fellowship support from University of Southern California. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or USC.

REFERENCES

1. Krause, A. 2008. *Optimizing sensing*, Ph.D. thesis, PhD thesis, Carnegie Mellon University.
2. Bian, F., D. Kempe, and R. Govindan. 2006. "Utility Based Sensor Selection," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, ACM, New York, NY, USA, IPSN '06, p. 11–18.
3. Krause, A. and C. Guestrin. 2007. "Near-Optimal Observation Selection Using Submodular Functions," in *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI Press, AAAI'07, ISBN 9781577353232, p. 1650–1654.

4. Ostachowicz, W., R. Soman, and P. Malinowski. 2019. “Optimization of sensor placement for structural health monitoring: a review,” *Structural Health Monitoring*, 18(3):963–988.
5. Bopardikar, S. D., O. Ennasr, and X. Tan. 2019. “Randomized sensor selection for nonlinear systems with application to target localization,” *IEEE Robotics and Automation Letters*, 4(4):3553–3560.
6. Hinson, B. T. 2014. *Observability-based guidance and sensor placement*, Ph.D. thesis, University of Washington.
7. Papadimitriou, C. 2004. “Optimal sensor placement methodology for parametric identification of structural systems,” *Journal of sound and vibration*, 278(4-5):923–947.
8. Yi, T.-H., H.-N. Li, and X.-D. Zhang. 2015. “Health monitoring sensor placement optimization for Canton Tower using immune monkey algorithm,” *Structural Control and Health Monitoring*, 22(1):123–138.
9. Mao, H., M. Alizadeh, I. Menache, and S. Kandula. 2016. “Resource management with deep reinforcement learning,” in *Proceedings of the 15th ACM workshop on hot topics in networks*, pp. 50–56.
10. Mirhoseini, A., A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, et al. 2020. “Chip placement with deep reinforcement learning,” *arXiv preprint arXiv:2004.10746*.
11. Hansen, N. and A. Ostermeier. 2001. “Completely Derandomized Self-Adaptation in Evolution Strategies,” *Evolutionary Computation*, 9(2):159–195.
12. Hansen, N. 2016. “The CMA Evolution Strategy: A Tutorial,” *arXiv prepring arXiv:10.48550*.