Expanding Reinforcement Learning Modeling Capabilities in Emergency Supply Distribution via Action Masking

Rudy Milani[1], Joshua Arnold[1], Maximilian Moll[1], Stefan Pickl[1]

[1] Department of Computer Science, University of the Bundeswehr Munich, Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
Email: rudy.milani@unibw.de

Abstract: Mitigating post-disaster human suffering through the provision of emergency resources is a challenging problem from a logistical standpoint. Finding optimal distribution strategies can be a challenging task. In this paper, we apply action-masked Reinforcement Learning to a novel formulation of the problem of emergency resource distribution within the context of disaster relief logistics. Specifically, we propose a solution approach for the scenario of supplying several locations of resource consumption via a single logistics hub. Our main contributions are twofold. Firstly, we extend prior work by defining a more complex mathematical optimization problem including multiple constraints through which we induce spillover effects over multiple time steps in the system dynamics. Secondly, we employ action masking in our reinforcement learning approach to help the agent avoid taking newly generated invalid actions. Furthermore, we compare one-step and two-step greedy heuristics with the action-masked version of Q-learning in a variety of simulated scenarios. Results confirm the usability of action-masked Reinforcement Learning even though the one-step greedy approach achieves the best performance-time ratio.

Keywords: Reinforcement Learning; Machine Learning; Humanitarian Logistics; Disaster Relief; Emergency Resource Allocation

1. Introduction

Alleviating the negative impact of natural or man-made disasters on the affected communities requires significant logistical effort in their immediate aftermath (Cozzolino, 2012). Allocating basic necessities such as water, food and medicine within a reasonable amount of time, given constrained supplies and means of transport can be a challenging task. The research community has coined the term *humanitarian logistics (*Cozzolino, 2012; Klumpp et al., 2015) to describe this logistical effort to mitigate post-disaster human suffering.

Prior work has discussed and solved a wide range of different formulations of the emergency resource allocation problem. For instance, (Jain and Bharti, 2023) aim to find optimal assignments of response units to emergency locations. (Wang et al., 2022) on the other hand, propose a more all-encompassing approach, i.e., they start from the path planning phase between the logistics hub and the final destinations. Other works such as (Fan et al., 2022) assume that the paths and associated transportation costs are given. Their problem definition will be extended in our work.

Recent years have seen a surge of interest in all areas of machine learning, including Reinforcement Learning (RL) (Schrittwieser, J. et al., 2020). RL's increasing popularity can be attributed to its effectiveness at solving sequential decision problems, which sit at the core of a variety of problems in logistics and operations research in general (Powell, 2022). In the following section, we will first briefly introduce RL and then discuss how (Fan et al., 2022) define their interpretation of the humanitarian logistics problem. Then, we will present our extensions to their optimization task, highlighting particular tricks and challenges in applying RL to such problems. Finally, we will present our solution approach to the aforementioned extended problem.

2. Background

2.1. Reinforcement Learning

Essentially, we are aiming to solve a sequential decision problem which can be modeled as a Markov Decision Process (MDP) (Sutton and Barto, 2018). MDPs are discrete time stochastic control processes. We can formalize an MDP as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is the state space, $\mathcal{T}: \mathcal{S} \times \mathcal{A} \to \mathcal{S} \times \mathcal{R}$ is the transition function and $\gamma \in [0,1)$ is a discount factor. After choosing an action $\mathrm{a}_t \in \mathcal{A}$, where $t \in \mathbb{N}_0$ denotes the time step, the agent enters a new state s_{t+1} and receives a reward r_{t+1} according to $\mathrm{s}_{t+1}, \mathrm{r}_{t+1} = \mathcal{T}(\mathrm{s}_t, \mathrm{a}_t)$. The agent's goal is to choose a sequence of actions that maximizes the expected return $\mathbb{E}[G_t]$, where $G_t \coloneqq \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1}$ (Sutton and Barto, 2018). We can easily see how the discounting factor γ can be set to place less focus on later rewards. Typically, the agent tries to learn a policy function $\pi: \mathcal{S} \to \mathcal{A}$ which suggests the optimal action in each state. A fundamental quantity in RL is the action value function $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t | s_t = s, a_t = a]$, which measures the expected future return, if, in state s, action a is chosen and policy π followed thereafter (Sutton and Barto, 2018). Q-learning (Watkin and Dayan, 1992), one of the most-used RL algorithms, attempts to learn the optimal action-value function through a combination of explorative experience and bootstrapping. The policy is derived by choosing the action that maximizes the obtained function. This fundamental setting, however, can only be applied to small and finite state and action spaces. Thus, DQN (Mnih et al., 2015) approximates the action-value function by a neural network and adds some more details to stabilize training.

2.2 Related Work

Finding optimal strategies for the humanitarian supply distribution problems are well-known studied challenges. However, the opportunity to consider machine learning-based approaches has not been extensively explored, in particular, in the case of Reinforcement Learning. On this thread, Fan et al. (2022) propose an RL-based approach to solve the problem of allocating a limited number of resources from a single logistics hub to a set of *N* distinct locations where these resources are consumed. The authors focus on the 72-hour time window after the disaster, as they propose that it is the most critical phase for emergency response (Fan et al., 2022). This time frame is partitioned into T sub-periods of equal length, i.e. $t \in \{1, \dots, T\}$. During each of these time steps, Fan et al. (2022) aim to allocate resources up to some available amount C which represents the resource capacity of the logistics hub. Sending, at time step t, some resources $a_t^i \in \mathbb{N}$ from the hub to location i is associated with some cost c_i. Fan et al. (2022) trace the evolving state of the system by assigning each location i with a state variable $S_t^i \in \mathbb{Z}$, which denotes the amount of demand at location i at time t. Each state can decrease via the allocation of new resources and increase via the inherent resource demand D of the location $S_t^i = S_{t-1}^i - a_t^i + D$. Fan et al. (2022) consider three distinct types of costs, namely *accessibility*, *deprivation* and *unfairness*. Fan et al. (2022) summarize their optimization objective as

$$min \; \underbrace{\xi_1 \sum_{i=1}^{N} \sum_{t=1}^{T} c_i \, a_t^i}_{\text{accessibility cost}} + \underbrace{\xi_2 \sum_{i=1}^{N} \sum_{t=1}^{T} \Gamma(S_t^i)}_{\text{deprivation cost}} + \underbrace{\xi_3 \sum_{i=1}^{N} \Gamma(S_{T+1}^i)}_{\text{unfairness cost}}, \qquad (1)$$

where the authors model the cost of accessibility by assuming that there is some $c_i \in \mathbb{N}$ associated with the allocation of each unit of resource to location i. Hence, the total accessibility cost can be written as $\sum_{i=1}^{N} \sum_{t=1}^{T} c_i \, a_t^i$. Furthermore, Fan et al. (2022) have defined $\Gamma(S_t^i) = e^a(e^{bL} - 1)(e^{bL})^{S_t^i}$ for $S_t^i \geq 0$ and $\Gamma(S_t^i) = 0$ otherwise, where a and b are the deprivation parameters, and L is the length of a single time period. Intuitively, the costs of deprivation and costs of unfairness counteract one another, as the former penalize under-delivering while the latter penalize over-delivering resources to locations. Finally, the ξ_i are scalar weighting factors. In the following, we will define our extension to this problem definition.

3. Methodology

3.1. Optimization Problem Formulation

We formulate our resource allocation problem via an optimization model as follows:

$$\begin{aligned}
&min && \sum_{i=1}^{N}\sum_{t=1}^{T} c_i \mathbb{1}_{\{a_t^i>0\}} + \Gamma\left(S_t^i\right) && \text{(2a)}\\
&s.t. && S_t = S_{t-1} + a_t - D && \text{(2b)}\\
& && C_t = C_{t-1} - \sum_{i=1}^{N} a_{t-1}^i + I_c && \text{(2c)}\\
& && a_t \leq F && \text{(2d)}\\
& && \sum_{i=1}^{N} a_t^i \leq C_t && \text{(2e)}\\
& && -M_S \leq S_t^i \leq M_S && \text{(2f)}\\
& && 0 \leq C_t \leq C && \text{(2g)}
\end{aligned}$$

for t = 1,2,...,T. Similarly to Fan et al. (2022), we aim to minimize transportation costs. However, we assume that there is no additional cost associated with transporting more units of resource to i. Intuitively, this can be interpreted as saying that once we decide to drive a truck to i for transporting one unit of resources, adding another unit of resources to this delivery brings no additional cost. Hence, we can rewrite the allocation cost as in the first part of Equation 2a, where $\mathbb{1}_{a_t^i>0}$ simply is an indicator function that is equal to 1 if $a_t^i > 0$ and else is equal to 0. Furthermore, we absorb costs of deprivation and unfairness into one term, namely into the second part of Equation 2a. Here, we redefine Γ such it holds that $\Gamma\left(S_t^i\right) = 0$ for $S_t^i \geq 0$, whereas else we require that $\Gamma\left(S_t^i\right) = e^a(e^d - 1)e^{-dS_t^i}$, with $d = bL$ and a as in the previously defined case.

Note, that we have changed the semantics of the state encoding from the original meaning *demand of i at t* in (Fan et al., 2022) to the new meaning of *supply at location i at time t*. Thus, the state updates are now described by Equation 2b. Additionally, we introduce a constant $I_c \in \mathbb{N}$ that models an incoming stream of resources to the logistics hub that arrives at each t. We assume that at each time step, we can allocate the sum of the remaining capacity from the last time step and the aforementioned fresh resource supply. Thus, we propose to reformulate this aspect as Equation 2c. Note that, in Equation 2g, it holds that $0 \leq C_t \leq C$. We restrict the action space by requiring Equation 2d, where $F \in \mathbb{N}^{\mathrm{N}}$ is a constant vector representing the limitation of supplies that can be sent to each location. Moreover, we introduce action masking for the agent via Equation 2e. Lastly, we added the constraint in Equation 2f in order to limit the possible configurations of the status S_t available to a maximum value $M_S \in \mathbb{N}^N$. In this way, our method could lend itself to scenarios of highly limited amounts of perishable resources, as we penalize both the under- and over-supply of locations as well as consider the arrival of fresh resources to the logistics hub at each time step. In the next subsection, we present the MDP formulation considered for the experiment developed.

3.1. MDP Formulation

In order to solve our problem with RL, we have to define three fundamental building blocks of the algorithm: (i) the *state space*, (ii) the *action space* and (iii) the *reward function*. Firstly, we define the states of our MDP as the vector of the status and the remaining capacity at time t as $s_t \coloneqq (S_t, C_t)$. Moreover, it can also be expressed with all the multiple components referring to each individual location i as $s_t = (S_t^0, \dots, S_t^N, C_t)$. This allows us to summarize all information necessary to take the correct action.

It is important to remark that the dimension of the state space is exponentially growing with respect to N. In particular, we have that $|\mathcal{S}| = (C + 1)(2M_S + 1)^N$. The same problem poses itself for the action space in the case that we model the action performed at time t as $a_t = (a_t^1, a_t^2, \dots, a_t^N)$. Combined with the constraint in Equation

2d, we are able to restrict the action space to $|\mathcal{A}| = \prod_{i=1}^{N}(F_i + 1)$. However, the set of admissible is much smaller. In fact, we have the dynamic updating of the upper capacity of resources that we can supply, represented in Equation 2e. Nonetheless, we consider a static action space in order to keep the problem simple and we solve this issue using action masking (Huang and Ontañón, 2020). This strategy is a well-known methodology in Reinforcement Learning, applied to complex games where multiple rule sets make possible actions feasible (Vinyals et al., 2017). Intuitively, action masking removes all actions from the action space of the agent that are invalid with respect to some constraints. In our case, we are taking out the possible actions that sum up to a higher value than the actual capacity, hence satisfying Equation 2e. Moreover, it is also possible to consider the action masking in order to train just a single RL agent for solving the problem for large size scenarios. In fact, by fixing the values relative to the areas not considered in smaller situations to 0, we can still consider the aforementioned model, masking the invalid actions that would involve the supply of not present areas. In this way, the training time could be extremely decreased.

Finally, we define the reward function our agent aims to maximize. In our case, the selection is trivial since the objective function can already model the most of the environment. The only change is in the sign associated with Equation 2a. In this way, we obtain that the reward as $r_{t+1} = -\left(\sum_{i=1}^{N} c_i \mathbb{1}_{a_t^i > 0} + \Gamma\left(S_{t+1}^i\right)\right)$. We have now defined all technical aspects fundamental to understanding the framework considered. In the following section, we will discuss the results of our simulation.

4. Results and Discussion

In this section, we assess the performance of our Q-learning approach by comparing it to two heuristic benchmarks, i.e., a one-step and a two-step greedy algorithm. The experiments were conducted on a server with 256 (2 x 64 +Hyperthreading) cores, 2x AMD EPYC 7763 CPU and 1007Gb RAM.

First, we need to declare the instances that have been considered for the test phase. In particular, we focus our attention on three possible values for the number of areas to supply, $N = \{3,4,5\}$, and four possible capacity limits for each of those values. In detail, we consider for $N = 3$ the values $C = 4{,}5{,}6{,}7$, for $N = 4$, $C = 5{,}6{,}7{,}8$ and lastly for $N = 5$ only $C = 6$. The maximum time step was always fixed at $T = 6$, as well as for the increase of the capacity $I_C = 3$, and the limits of the status of each location i, $M_S = 3$. The initialization of the states is performed considering a uniform distribution between the bounds of both the status and the capacity. Depending on the dimension of the problem, we constructed similar constant vectors for the demands and the capacity of the supplies. For the former, we created a vector that is equal to 2 in the even positions and 1 in the odd ones, i.e., $D = (1,2,1,\ldots)^T$. Using the same idea, we constructed $F = (2,3,2,\ldots)^T$, where we used 2 for the odd indexes and 3 for the even. Lastly, we have to declare the parameters associated with the reward function. Principally, the cost vector is generated starting with a fixed amount and, then, constantly adding an increment value for each new location. Therefore, the singular costs follow the rule $c_i = 200 + 50i$. Finally, for the deprivation/unfairness function, we used $a = 3$, and $d = 1.5$.

Under these conditions, we compared the Q-learning method with the one-step and two-step greedy algorithms (Efroni, Y. et al., 2018). These methodologies are simple heuristics based on the idea of taking the action that maximizes the reward obtained in one or two steps, respectively. Therefore, they have to first evaluate all the possible combinations of actions in order to find the correct estimate. However, this approach is extremely expensive since it must iterate the same analysis at multiple time steps. Thus, it is not optimal in terms of time consumption to search for the exact solution. For Q-learning instead, we can directly fix an amount of iterations that we want to consider and train until that point. In this way we can limit the necessary time needed at cost of not finding the best policy. For our experiments, we trained the Q-learning algorithms using a fixed set of hyperparameters: learning rate $\alpha = 0.01$, discounting factor $\gamma = 1$, starting exploration value $\epsilon_0 = 1$, minimum exploration value $\epsilon_m = 0.01$, subtractive decreasing exploration factor $\Delta\epsilon = \frac{1}{L}$, where L represents the number of iterations used for the training. The latter parameter is chosen accordingly to the size of the problem: for $N = 3$ we used $L = 10^6$, for $N = 4$ we considered $L = 5 \times 10^6$, and lastly, for $N = 5$, $L = 10^7$.

These methodologies are tested over 100 seeded episodes. The results and the time needed for the training are shown in Table 1. Here, the gap is calculated taking into account the best-performing algorithm, i.e., the two-step greedy, and averaging over all the episodes. From these results, we can notice how the Q-learning algorithm is able to outperform both the greedy approaches for the small set environments, i.e., $N = 3$ and $C = 4$ and 5. However, it starts to get worse when the state and action spaces are larger. In fact, in those cases, it has to visit a higher number of possible states, thus, it is more probable to not find the correct policy to perform. Moreover, the training time of the Q-learning methodology strictly depends on the number of episodes considered. Nonetheless, the longer the training the best are the results achieved. Particularly remarkable are also the performances of the one-step greedy strategy. Indeed, it is the methodology that performs in the best way in terms of quality/time ratio. Obviously, even though there is always a gap with the approach that looks one step further, the times are still increasing in a much slower ratio.

Params.			**Two-Step Greedy**		**One-Step Greedy**		**Q-Learning**	
N	C	T	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
3	4	6	0.0%	15.90	8.50%	0.46	-5.00%	548.13
3	5	6	0.0%	37.28	8.03%	0.79	-0.17%	581.31
3	6	6	0.0%	69.34	11.17%	1.20	3.96%	575.82
3	7	6	0.0%	107.90	27.12%	1.65	15.05%	606.91
4	5	6	0.0%	1116.22	5.91%	11.48	10.97%	3142.67
4	6	6	0.0%	2345.01	4.51%	18.97	12.37%	3307.43
4	7	6	0.0%	4327.89	4.81%	29.31	16.48%	3432.32
4	8	6	0.0%	8974.58	2.66%	44.33	28.52%	3583.66
5	6	6	0.0%	62826.83	7.38%	276.27	33.40%	7169.24

Table 1. Results obtained on 100 episodes.

This is principally due to the explosion of all the possible combinations of actions that can be tested, as can be clearly visible by looking at the training times of the two-step greedy. This quick increase in the dimension of the environment is a major computational problem. Furthermore, we also tested a DQN approach to the problem knowing the high performance capabilities of deep reinforcement learning methodologies. However, we were able to set up the experiment only for the smallest value of $N = 3$ since the memory of the GPU (NVIDA A40 with 138204MiB VRAM) was not enough to store all the data generated. Moreover, the results obtained after 4 hours of training led only to bad results (a gap of 600% compared to the two-step greedy). Therefore, we avoided further exploring this solution.

5. Conclusion

In this paper, we extended an emergency supply allocation problem by including a dynamic capacity constraint to the resources and compared a Reinforcement Learning algorithm with two greedy heuristics in solving the aforementioned optimization task. From the simulation generated, we noticed how Q-learning is able to outperform both the greedy approaches for small instances. Therefore, in those scenarios, it is advisable to apply a Reinforcement Learning methodology. However, we need to train it for a longer time compared to the other heuristics considered in this paper. For larger instances, the best solutions, in terms of minimizing the objective function, is obviously the two-step greedy. Nonetheless, the computation time is remarkably high. Therefore, if we want to obtain satisfying results in a decent time (order of minutes), the choice is the one-step greedy.

Fundamental for this experimentation is the action masking strategy, that allowed us to consider a stationary action set. Moreover, a possible direction for future study is the analysis of the results obtained in smaller instances by the masked version of the larger Q-learning agent. Furthermore, it would be interesting to test the

Reinforcement Learning algorithms in contexts where the updating variables are stochastics. In those situations, they could provide interesting results if compared to the classical approaches.

Acknowledgements

Rudy Milani is funded by dtec.bw—Digitalization and Technology Research Center of the Bundeswehr project RISK.twin. dtec.bw is funded by the European Union—NextGenerationEU.

References

Cozzolino, Alessandra (2012). "Humanitarian Logistics", In: *SpringerBriefs in Business*, doi: 10.1007/978-3-642-30186-5_1

Efroni, Y., Dalal, G., Scherrer, B., & Mannor, S. (2018, July). Beyond the one-step greedy approach in reinforcement learning. In International Conference on Machine Learning (pp. 1387-1396). PMLR.

Fan, Junchao et al. (2022). "DHL: Deep reinforcement learning-based approach for emergency supply distribution in humanitarian logistics". In: *Peer-to-Peer Networking and Applications* 15.5, pp. 2376–2389. issn: 1936-6450. doi: 10.1007/s12083-022-01353-0. url: https://doi.org/10.1007/s12083-022-01353-0.

Huang, S. and Ontañón, S. (2020), "A closer look at invalid action masking in policy gradient algorithms", arXiv preprint arXiv:2006.14171

Jain, Sehej and Kusum Kumari Bharti (2023). "A combinatorial optimization model for post-disaster emergency resource allocation using meta-heuristics". In: *Soft Computing* 27.18, pp. 13595–13611. issn: 1433-7479. doi: 10.1007/s00500-022-07176-8. url: https://doi.org/10.1007/s00500-022-07176-8.

Mnih, V. et al. (2015). "Human-level control through deep reinforcement learning". *Nature* 518, 529–533. https://doi.org/10.1038/nature14236

Powell, Warren B. (2022). "Sequential Decision Analytics and Modeling: Modeling with Python". now publishers.

Schrittwieser, J. et al. (2020). "Mastering atari, go, chess and shogi by planning with a learned model". Nature, 588(7839), 604-609.

Sutton, Richard S. and Barto, Andrew G. (2018). "Reinforcement Learning*:* An Introduction". Cambridge, MA, USA: A Bradford Book. isbn: 0262039249.

Vinyals et al. (2017), "Starcraft ii: A new challenge for reinforcement learning", arXiv preprint arXiv:1708.04782

Wang, Feiyue et al. (2022). "Multiobjective Emergency Resource Allocation under the Natural Disaster Chain with Path Planning". In: *International Journal of Environmental Research and Public Health* 19.13. issn: 1660-4601. doi: 10.3390/ ijerph19137876. url: https://www.mdpi.com/1660-4601/19/13/7876.

Wang, Yanyan and Sun, Baiqing (2022). "Multiperiod Equitable and Efficient Allocation Strategy of Emergency Resources Under Uncertainty". In: *International Journal of Disaster Risk Science* 13.5, pp. 778–792. issn: 2192-6395. doi: 10.1007/ s13753-022-00437-y. url: https://doi.org/10.1007/s13753-022-00437-y

Watkins, C.J.C.H., Dayan, P (1992). "Q-learning". *Mach Learn* 8, 279–292. https://doi.org/10.1007/BF00992698

Van Wassenhove, LN (2006). "Humanitarian aid logistics: supply chain management in high gear". Journal of the Operational Research Society, 57:5, 475-489, DOI: 10.1057/palgrave.jors.2602125