Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

Bounty Hunters as Dynamic Traveling Repairmen

Drew Wicke dwicke@gmu.edu
Ermo Wei ewei@cs.gmu.edu
Sean Luke sean@cs.gmu.edu

Technical Report GMU-CS-TR-2018-2

Abstract

Vehicle routing problems such as the multiagent Dynamic Traveling Repariman Problem (DTRP) are of interest to many fields and of increasing practical importance in light of advances in autonomous vehicles. DTRP is NP-hard, making approximation methods attractive. However current heuristic approaches do not adequately consider issues special to DTRP, such as fairness and variance, discontiguous-space scenarios, or approaches to equitably partitioning the task space. We tackle this problem in a novel way, using a multiagent task allocation technique called *bounty hunting*. In bounty hunting, agents compete to perform tasks non-exclusively in return for reward, and rapidly learn which agents are more adept at various tasks than others, divvying up the task space. We demonstrate that bounty hunting can perform efficiently in discontiguous environments, and can improve the bias and variance of the system while minimally affecting average waiting time. We show that Bounty Hunting Pareto dominates the current state-of-the-art heuristic, and is particularly good in large-scale scenarios.

1 Introduction

In this paper we discuss a new approach to the distributed or multi-agent version of the *dynamic traveling repairman problem* or DTRP [1]. In the DTRP, one or more agents are tasked to travel to and from various locations in order to service customers. The goal is to minimize the average waiting time of the customers. While abstract, the DTRP is applicable to a wide and increasing range of real-world problems, notably in routing, autonomous vehicles, and logistics.

The DTRP is NP-hard and so much of the literature has focused on heuristic approximation methods. Surprisingly, one of the most popular and best-performing single-agent-case solutions is to simply follow a *nearest neighbor* policy [2]. In the distributed case, it has been proven that by equitably partitioning the space into k regions, each assigned to a unique agent, and by following some optimal single agent policy, the overall system will be near optimal in general [14].

Unfortunately very little research has been done in building equitable partitions. Current approaches attempt an equitable partition of the space by running a distributed gradient descent algorithm on the weights of a power diagram [13]. These methods assume that the agents are resources to be allocated to the generated regions, and that the agents share internal information between themselves in order to create the equitable partitions. This may fail if not all the agents are following the same algorithm, or are otherwise unwilling or unable to share information.

Additionally, these methods assume a contiguous space, and while some of the current algorithms function in other spaces, current theory does not address this situation. However in many real world scenarios there will be spaces where no tasks are generated, creating discontiguous task regions.

Finally, and critically, these methods have only focused on the *efficiency* of the system (in the form of average waiting time) but not the variance, which can contribute to the perception of a longer waiting time [9]. Unfortunately work on studying variance even in the single agent case has been limited [8]. Furthermore, *fairness* (for example, the order of task completion) has not been studied in this setting and no formal fairness metric has been proposed. However, fairness is an important concern in many similar areas, such as in scheduling queues [19]; and in real-world scenarios to which the DTRP may be applied, particularly involving humans.

Bounty Hunting We will demonstrate the use of *bounty hunting* as an effective alternative heuristic solution to the DTRP. Bounty Hunting is a novel counterpart to the use of auctions in multiagent task allocation [16] in which a *bail bondsman* offers an ever-increasing *bounty* (or reward) for various tasks up for grabs, and multiple *bounty hunters* compete to complete these tasks. The

bounty on a task is rewarded only to the bounty hunter who completes it first. There is no task exclusivity: multiple bounty hunters can commit to the same task at the same time. Although, tasks can not be simultaneously serviced by more than one agent. For purposes of the DTRP, this means that our bounty hunting variation will not partition the space at all, unlike previous methods.

Bounty hunting only works if each agent can adapt so as to learn which tasks are worth pursuing. At that point they will have largely ceded tasks to one another, effectively (in the DTRP case) creating a loose partition of the space. However this partition is dynamic, soft, and fluid, allowing for flexibility that the hard-partitioning methods cannot provide. Furthermore this approach works better if the bounty hunters are equipped with the ability to *signal* to other bounty hunters that they intend to work on specific tasks.

The bail bondsman gradually increases the bounty on an outstanding task until it is completed. We argue that the rate at which this bounty increases can act as a tuning knob for the variance in the average waiting time of the tasks, and will show how changing this rate affects both efficiency and fairness, allowing the designer to tune the system for the task at hand.

In this paper we will begin by describing the problem and past approaches to solving it. We will then present our version of the bounty hunting model and signaling method, and show that in the single-agent case it can be straightforwardly made equivalent to the nearest-neighbor method in terms of efficiency. We define several ways to measure both the efficiency and fairness of the system, and then perform a number of experiments to show the advantages of bounty hunting. Our experiments will first show how bounty hunting can be equivalent to nearest-neighbor in the single-agent case, while being tunable for both efficiency and fairness. We will then move to the multiagent case, showing how our partition-free approach can significantly outperform existing hard-partitioned distributed methods.

2 The Multiagent Dynamic Traveling Repairman Problem

In the multiagent DTRP we define a set $\mathcal{M}$ of m agents, each with point locations at time t, as $H(t) = \langle h_1(t), ..., h_m(t) \rangle$. Each task i has a location l_i and an identical and independently distributed service time with mean $\bar{s}$. Tasks are generated by a Poisson point process with a Poisson rate λ and a location uniformly distributed on a convex space $\mathcal{G}$ of area G. The *load factor,* or the percent of time an agent spends servicing a task, is given by $\rho = \frac{\bar{s}\lambda}{m}$. The system is defined to be in heavy load when $\rho \to 1$ and in low load when $\rho \to 0$. Agents move at constant velocity v within G. Tasks are removed after an agent arrives to the task and services it. An agent may work on only one task at a time. The time from the arrival of the task to its completion is T_i. The system objective is to minimize the average wait time of the tasks, $T = \lim_{i \to \infty} E[T_i]$.

Work has been done to characterize the expected waiting time of the tasks for a variety of single and multiple vehicle algorithms. In [1] it was shown that the average waiting time, in heavy load conditions, is bounded below as:

$$T \geq \frac{\lambda G}{4v(1-\rho)^2}$$

Further, the bound may be written, for particular γ that is determined for a specific policy μ, as:

$$T \sim \gamma_\mu^2 \frac{\lambda G}{v(1-\rho)^2}$$

Values for γ have been analytically provided for a number of different policies [2]. However, the nearest neighbor policy has no fixed value for γ and it has only been possible to experimentally determine its value [4].

The bounds in heavy load have been extended to the multiagent setting [3]. There are two main ways of dividing up the problem between the agents. First, the Poisson process can be equally divided between the agents. In this case, it is easy to see that:

$$T \sim \gamma_\mu^2 \frac{\lambda G}{mv^2(1-\rho)^2}$$

However, this method does not scale as the number of agents increases. Another popular approach has been to equitably partition the space, assign each agent to a region, and have each agent service the tasks in its region using a single agent policy. In this case we have:

$$T \sim \gamma_\mu^2 \frac{\lambda G}{m^2v^2(1-\rho)^2} \quad (1)$$

Because both the area and the Poisson point process are divided equally between the agents, the method scales with the number of agents [3].

3 Related Work

The DTRP has a number of practical applications including fleet routing of repair and utility vehicles [13, 5]. Bertsimas and van Ryzin formalized a number of algorithms for solving the problem in the single vehicle scenario [1]. They showed that the DTRP is related to queuing theory and provided bounds for the average wait time in both heavy and light load situations. Further, they experimentally found the Nearest Neighbor approach outperforms a TSP approach, which has theoretical guarantees for γ that NN does not. They state that NN is not *fair* because it does not follow a First Come First Serve (FCFS) policy, and does not repeatedly return to a single location [2].

Single agent DTRP has been further studied in various conditions and settings. For example, a new approach, PART-n-TSP, reduces the variance of the waiting time in mid-range loads [8]. The DTRP has also been extended to include deadlines for tasks, with upper bounds on competitive ratios given for certain algorithms [10]. DTRP has also been studied for networks rather than Euclidean space [20].

DTRP is normally extended to the multi-vehicle (multi-agent) case by partitioning the space into regions, and running single vehicle policies, one vehicle per region [3]. There exists a generalized form of the problem with demand locations and service times based on arbitrary continuous distributions [4]. Equitably partitioning the space has also been generalized by performing gradient descent on the weights in power diagrams [13, 14].

Dynamic multiagent task allocation bears many similarities to the multiagent DTRP and methods in this area may prove fruitful. Such approaches include auctions [6] and bounty hunting [16, 18, 17]. However to date market, auction, or economic based approaches (in which bounty hunting falls) have not been studied in the DTRP. However, there has been work done in a related problem, the k-TRP, where tasks are not dynamically generated and the agents may bid for the tasks in an auction [11].

Many of the approaches to the DTRP have been based on scheduling disciplines such as FCFS (First Come First Served) and its SJN (Shortest Job Next) equivalent, namely Nearest Neighbor. Bounty hunting is closely related to another scheduling discipline, Highest Response Ratio Next (HRRN) [7]. This method works by scheduling jobs with the highest priority where the priority is defined as:

$$Priority = \frac{Waiting\ Time}{Estimated\ Run\ Time} + 1$$

This ratio has since been used as a slowdown metric, and is very similar to the bounty metric used in [16]. Recently the HRRN has been extended to be preemptive, which is again similar to bounty hunting with task abandonment [15, 18].

4 Methods

4.1 Bounty Hunting Model

The bounty hunting model has previously been studied and formally defined for the dynamic multiagent task allocation problem [16, 18].

Bondsman The bondsman, who may be distributed among the tasks, acts to set the base bounty and the bounty rate for the tasks. The base bounty, B_0 is the bounty assigned to the task at time step zero, and the bounty rate R is the rate at which the bounty rises with time. Therefore, the bounty associated with a particular task at time step t is $B(t) = B_0 + Rt$. The bounty rate may be a function; however, we restrict ourselves to a constant $R \geq 0$.

When setting the base bounty, the bondsman must consider the resource cost of the bounty hunters. One such resource is fuel. The fuel resource costs bounty hunters C_f per unit of travel. Therefore, the base bounty should be conservatively set to $B_0 > C_f \frac{\|G\|}{v}$ where $\frac{\|G\|}{v}$ is the maximum number of units a bounty hunter may travel.

In addition, the bondsman must set the bounty rate R. To provide guidance on setting the bounty rate we assume the bounty hunters choose to pursue tasks so as to maximize their bounty per time step and are not hyperstrategic (assumed in previous literature in bounty hunting [16]). We recognize that the nearest neighbor algorithm in the single agent case has been experimentally shown to produce near optimal results [1]. Therefore, it would be ideal to motivate the bounty hunters to follow a nearest neighbor policy.

We let $I(t)$ be the set of tasks available at time t. Assume we have a single bounty hunter whose location at time t is $h_1(t)$, and that the distance between task $i \in I(t)$ and the agent is defined to be $d_i = \|l_i - h_1(t)\|$.

Definition 1. *Nearest Neighbor Policy*

$$\underset{i \in I(t)}{\operatorname{argmin}}\, d_i \tag{2}$$

Definition 2. *Greedy Bounty Hunter Policy*

$$\begin{aligned} &\underset{i \in I(t)}{\operatorname{argmax}}\, U_i(t) \\ &U_i(t) = \frac{B(t) - C_f d_i}{d_i + \bar{s}} \end{aligned} \tag{3}$$

To motivate the bounty hunters to follow the nearest neighbor policy we make the following proposition.

Proposition 1. *Let $B_0 > C_f \frac{\|G\|}{v}$, $R = 0$, and let there be a single bounty hunter following the greedy bounty hunter policy, then the bounty hunter will be incentivized to service tasks with a nearest neighbor policy.*

To prove this we derive the nearest neighbor policy from the greedy bounty hunter policy. To simplify the equations we can assume without loss of generality that $v = 1.0$.

Proof. First, we convert the maximization problem of the bounty hunter to a minimization problem

$$\underset{i \in I(t)}{\operatorname{argmax}}\, U_i(t) = \underset{i \in I(t)}{\operatorname{argmin}} -\frac{B(t) - C_f d_i}{d_i + \bar{s}} \tag{4}$$

Now we assume that there are two tasks 1 and 2 that have utility $-U_1 < -U_2$. The agent will then choose to

vie for task 1 according to Equation (4). We prove below that by setting $B(t) = B_0$ for all tasks we have:

$$\begin{aligned} -\frac{B(t) - C_f d_1}{d_1 + \bar{s}} &< -\frac{B(t) - C_f d_2}{d_2 + \bar{s}} \\ \frac{B(t) - C_f d_1}{d_1 + \bar{s}} &> \frac{B(t) - C_f d_2}{d_2 + \bar{s}} \\ (B(t) - C_f d_1)(d_2 + \bar{s}) &> (B(t) - C_f d_2)(d_1 + \bar{s}) \\ B(t)d_2 - C_f d_1 \bar{s} &> B(t)d_1 - C_f d_2 \bar{s} \\ d_2 &> d_1 \end{aligned} \tag{5}$$

Therefore, the task with the minimum distance d_1 is chosen by the nearest neighbor policy. It follows that by choosing the task with the highest bounty per time step that the agents will choose the task that is closest to them and therefore are following the nearest neighbor policy:

$$\underset{i \in I(t)}{\operatorname{argmax}} U_i(t) = \underset{i \in I(t)}{\operatorname{argmin}} -U_i(t) = \underset{i \in I(t)}{\operatorname{argmin}} d_i \tag{6}$$

□

We have shown that the bounty hunters will follow a nearest neighbor policy when the bounty rate is zero and the base bounty is fixed. However, when the bounty rate is set to some value other than zero the bounty hunters will no longer follow a nearest neighbor policy. If the bounty rate is set to some positive value, greedy bounty hunters will be incentivized to follow a policy similar in nature to both the nearest neighbor and FCFS policies. We will study this behavior in terms of variance in the average waiting time and fairness with respect to a FCFS policy with experiments in Section 6.

Bounty Hunter Learning and Communication Here, we consider how the bounty hunters learn when to signal to the other agents that they are attempting a task. Bounty hunters decide what task to work towards at each time step, and may abandon a task that they are working on or traveling toward in order to undertake another task. They keep track of the distance to the task that they were traveling toward when they abandon the task, in order to develop an expected distance. This distance indicates the range at which it becomes more probable that the agent will complete the task without abandoning it. For each agent j the set of distances at which agents will abandon tasks is defined as D_j and the average distance is defined as:

$$E[D_j] = \frac{1}{|D_j|} \sum_{d \in D_j} d$$

The bounty hunters then use this value to determine when to signal to other agents that they are pursuing a task i (where l_i is the location of task i):

$$\|h(t) - l_i\| \leq E[D_j]$$

Specifically, when the distance between the agent and the task is less than or equal to the point of average task abandonment, the agent will signal to nearby agents their intention of completing the task i. Initially $E[D_j]$ is set to zero and only after agents abandon tasks do those agents signal. Signaling gives a more reliable indication of the agent's intentions while allowing the agents to have the freedom to abandon tasks.

The bounty hunters use these signals and past interaction with agents in order to determine the probability of successfully completing the task. Because the agents are operating in a dynamic environment, exponential averaging is used to learn this value:

$$Y_k \leftarrow (1 - \beta) Y_k + \beta y$$

where y is 0 if the agent did not complete the task when agent k completed the task, and $\beta = .99$. Regardless of outcome the agent will increase the probability of success with all agents using the same update again, this time with $\beta = .001$ and $y = 1.0$. This has the effect of making agents slowly more optimistic about beating other agents to tasks. Such an effect is useful when there is an agent that has signaled the task but has possibly stopped working [16].

We can then calculate the probability of successfully completing the given task, based on the agents that have signaled that task or nearby tasks. Let i_j be the task that agent j signaled and let the set of agents Z be defined as:

$$Z = \{\forall k \neq j | i_j = i_k, \|l_{i_k} - l_{i_j}\| < \|h_j(t) - l_{i_j}\|\}$$

Then we can calculate, using this set of agents, the probability of successfully completing the task:

$$\alpha = \prod_{k \in Z} Y_k$$

Using the above we can then calculate expected bounty received for task i for agent j:

$$U_i(t) = \alpha \left(\frac{B_i(t) - C_i}{\|h_j(t) - l_i\| + \tilde{s}} + R \right)$$

where the cost is defined as $C_i = C_f \|h(t) - l_i\|$, the price of fuel is C_f, the agent's current location $h(t)$, and the task location is l_i. The average service time for the job is learned through exponential averaging $\tilde{s} = (1 - \psi)\tilde{s} + \psi s$, where s is the service time of the finished job, and ψ is the learning rate (in our experiments $\psi = 0.05$).

When agents arrive to a task, they work on the task until they finish it. The set of available tasks is defined as $I(t)$. At each time step the agents calculate which task they will travel toward as $I^*(t)$ or, if they are working on a task, whether they continue to work on the task by:

$$I^*(t) = \underset{\forall i \in I(t)}{\operatorname{argmax}} U_i(t)$$

Therefore, the bounty hunters learn not to go after tasks other agents have signaled while maximizing their total expected bounty per time step.

4.2 Equitable Partitions Nearest Neighbor

For this method we split the area G into m equitable partitions and assign each agent to a partition. Because we consider uniformly distributed tasks in a square region we are able to manually partition the space. However, if the tasks are distributed by some other distribution in a convex space then the space may be equitably partitioned through a gradient descent algorithm of a power diagram as defined in [13]. Then, each agent follows the single agent nearest neighbor policy, as defined in Definition 1, with the ability to abandon tasks.

5 Evaluation Metrics

In order to evaluate bounty hunting as an adaptive method for the multiagent DTRP, we propose a number of metrics.

Average Waiting Time of Tasks

Definition 3. *Average Waiting Time*

$$E[T](t) = \frac{1}{|K(t)|} \sum_{x \in K(t)} T_x \tag{7}$$

$K(t)$ is the set of completed tasks at time step t and waiting time for some task x is $T_x = W_x + s_x$. W_x is the time the task waited until it was serviced and s_x is the time it took to be serviced.

The experimental system time can be modeled by Equation (1). However, the value for γ is not known and is thought to be analytically intractable for the nearest neighbor policy [4]. Therefore, we must experimentally determine the value. The value of γ will provide evidence to support the claim that although the bounty hunters are not explicitly equitably partitioning the space they are modeled by the same formula.

Variance

Definition 4. *Variance of the Average Waiting Time*

$$Var(t) = \frac{1}{|K(t)| - 1} \sum_{k \in K} (T_k - E[T](t))^2 \tag{8}$$

Where $K(t)$ is the set of tasks that have been completed by time step t and T_k is the waiting time for task k. Variance enables us to examine another dimension of the efficiency of the system. Additionally, as we will see in Section 6 it helps us to understand the role of the bounty rate.

No theoretical bounds have been established for variance. With variance, however, we are able to better experimentally demonstrate the usefulness of a bounty rate.

Fairness Another metric we are interested in is the fairness of the system. In many scenarios we may be willing to sacrifice efficiency in order to service tasks that have been waiting a long time. The FCFS policy has been claimed to be fair [1]. We base our fairness metric on a comparison of the agent's actual decision to the decision had the agent followed a FCFS policy.

Definition 5. *Fairness*

$$Fair(x) = W_x / W^* \tag{9}$$

Where x is the task that the agent has chosen, W_x is the time the task has been waiting at the moment of the decision, and the waiting time of the longest waiting task is W^*. Therefore, an agent's decision is considered to be completely fair when $\text{Fair}(x) = 1$ and the agent services the task that has been waiting the longest. The decision is completely unfair when $\text{Fair}(x) = 0$ and the agent services the task that has been waiting the least amount of time.

We can then obtain the average fairness of the system by averaging the fairness of all of the decisions:

$$E[Fair] = \frac{1}{|A|} \sum_{j \in A} \frac{1}{|K_j|} \sum_{x \in K_j} Fair(x)$$

Where we have the set of agents A and K_j is the set of tasks that agent j has completed.

We restate fairness in terms of bias, a common metric in machine learning algorithms. Bias is the distance from complete fairness to the expected fairness:

$$Bias = 1 - E[Fair]$$

The decision making is more biased when decisions are less fair and less biased when the decisions are more fair. Bias $\to 1$ when the decision function completes tasks in a last in first out fashion and Bias $\to 0$ when processing tasks in a FCFS manner.

Total Error We are interested in the trade-off between bias and variance and minimizing the total error, where the total error is defined as:

$$Total\ Error = Bias^2 + Var(t)$$

With this measure we are able to analyze the effect of different values for the bounty rate on the average wait time of the tasks within the system.

Total Average Bounty Available The total outstanding bounty available in the environment can act as a measure for the efficiency of the system. In the DTRP we are able to provide a bound on the outstanding bounty in the system based on the bound on the system time and Little's Law.

$$\mathcal{B} \sim B_0 N + RTN = B_0 T\lambda + RT^2\lambda \tag{10}$$

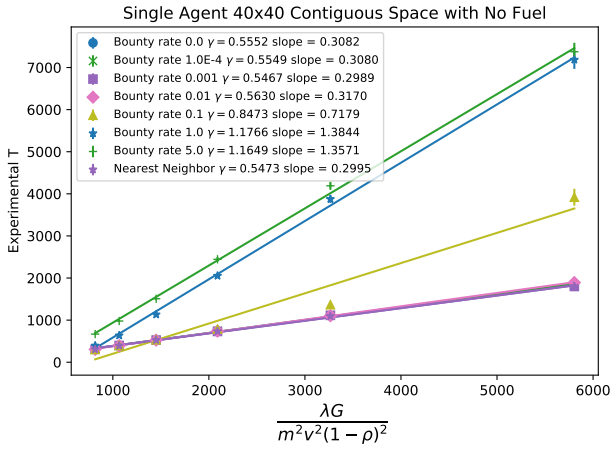

Figure 1: Contiguous Task Experiment: One agent 40×40 area with no fuel, 300,000 steps.

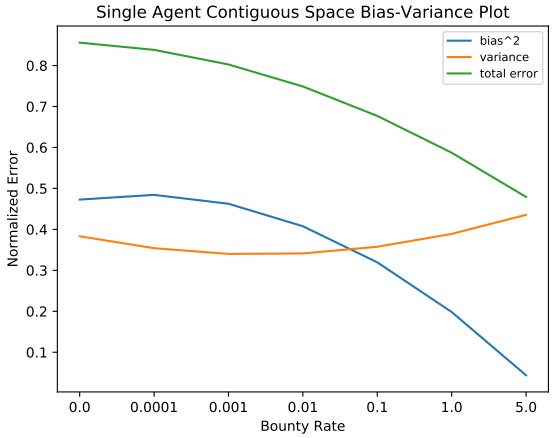

Figure 2: Contiguous Task Experiment: Bias, variance, and total error for one agent 40×40 area with no fuel, 300,000 steps.

where N is the number of tasks in the system on average, which can be estimated through Little's Law $N = \lambda T$. However, because the bounty hunters complete tasks in a nearest neighbor or nearest neighbor-like discipline, and the value for T is analytically intractable, we can only use this for known values of γ [4]. We are interested in studying this value in order to show that the amount of bounty in the system is bounded in terms of the average waiting time of the tasks.

6 Experiments

The tasks were generated in a continuous environment based on a Poisson point process with average time between tasks λ. Tasks were distributed uniformly randomly within the space defined by the experiment and the service time for each task was generated by a geometric distribution with a mean $\bar{s}$ specific to each experiment. The agents may overlap with other agents or tasks and moved with a velocity of $v = 0.7$. The agents traveled to the tasks and upon arrival serviced the task.

Agents were positioned at time step zero at a depot. Depots were locations in the environment where agents could purchased fuel or return to if no tasks were available to complete. For experiments where fuel was considered we set the unit cost of fuel $C_f = 1.0$ where the currency used was the bounty that the agent had obtained by completing tasks. In scenarios where fuel was used the agents had a fuel capacity of 3000 and started with a full tank along with a starting balance of 1000 units of bounty. Each time step that the agent moved in the environment would use a single unit of fuel. When the agent had only enough fuel to return to the depot the agent would refuel by returning and purchasing the max amount of fuel that they had currency for. It takes a single time step to refuel and we assume there was no additional waiting time if there was more than one agent refueling at the same depot. For all experiments we set the base bounty to a fixed $B_0 = 500$.

Table 1: Contiguous Task Experiment: Bounty ratio for one agent 40×40 area with no fuel, 300,000 steps; NN is the Nearest Neighbor approach.

	R							
$\bar{s}$	0.0	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1.0	5.0	NN
8.00	1.07	1.07	1.22	1.14	0.45	0.17	0.24	1.15
9.00	1.30	1.17	1.13	1.12	0.52	0.20	0.28	1.15
10.00	1.16	1.20	1.21	1.26	0.47	0.27	0.36	1.25
11.00	1.17	1.11	1.12	1.17	0.42	0.39	0.46	1.25
12.00	1.07	1.07	1.24	1.04	0.51	0.52	0.60	1.14
13.00	1.11	1.00	1.11	1.04	0.93	0.66	0.71	1.12

We tested the bounty hunting model in six different settings in MASON, a Java based multiagent discrete event simulator [12]. First, we looked at the single agent case with tasks in contiguous space and with a range of bounty rates. Second, we again have a single agent, but the tasks were in a discontiguous space. In both of these settings we examined the usefulness of the bounty rate and its effects on the system. Third and fourth, we considered two different four-agent settings with different values for λ and $\bar{s}$ but with the same value for ρ. Here we were interested in examining the effect of not explicitly partitioning the space into equitable partitions for the bounty hunting method. Fifth and sixth, we used a 64 agent system and focused on the scalability of the system, and discussed preliminary findings on the effect of fuel on the efficiency of the system. We left all simulations run for 300,000 time steps (except when noted).

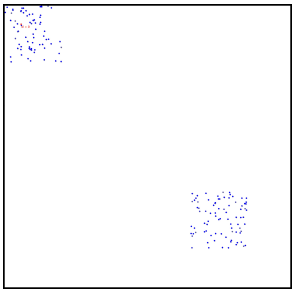

Figure 3: Discontiguous Task Space. One agent in the top left 40×40 area, after 600,000 time steps when $\bar{s} = 8$.

Table 2: Discontiguous Task Experiment: Average waiting time for the Nearest Neighbor approach and Bounty Hunting with a bounty rate of 5.0.

$\bar{s}$	Nearest Neighbor	Bounty Hunting
8.00	9305.95	2325.55
9.00	12211.07	3189.18
10.00	16086.59	4617.20
11.00	21221.22	7123.05
12.00	27554.38	11684.93
13.00	39723.68	21157.16

6.1 Single Agent

We examine two single agent environments in order to illustrate why the bounty rate is important.

Contiguous space For this experiment we verified that a bounty rate of zero is equivalent to the NN approach by comparing the value of γ. We then examined the effect of the bounty rate on the bias and variance. We compared the theory and actual total average bounty available in the system.

We had a single agent with start position located at the depot at (0,0), the center of the 40×40 environment. We set $\rho = 0.8125, 0.75, 0.6875, 0.625, 0.5625, 0.5$ based on $\lambda = \frac{1}{16} = .0625$ and mean job length $\bar{s} = 13, 12, 11, 10, 9, 8$. We set the bounty rate to $R = 0, 0.0001, 0.001, 0.01, 0.1, 1, 5$. We repeated the simulation 40 times for each value of ρ and bounty rate.

Results In Figure 1 we plot the experimentally obtained average waiting time T against $\frac{\lambda G}{m^2 v^2 (1-\rho)^2}$ for a range of bounty rates alongside the NN approach. We can see that the bounty hunting approach with zero bounty had a γ value within 1% of the nearest neighbor approach.

Next, we consider in Figure 2 how the bounty rate effected the bias and variance of the system. To plot this we averaged the bias and variance for each $\bar{s}$ and then normalized the values. As the bounty rate increased the bias decreased and the variance decreased and then increased, but the total error was minimized when the bounty rate R was 5. However, when the bounty rate $R \geq 0.1$, the expected waiting time increased. Therefore, there was a trade-off between average waiting time, and the bias and variance.

Each entry in the bounty table in Table 1 is the ratio of the total bounty in the system at time step 300,000 averaged over 40 trials to $\mathcal{B}$. We calculated $\mathcal{B}$ based on the values for γ found in Figure 1. Values greater than one meant that $\mathcal{B}$ underestimated the bounty in the system and when the ratio was less than one $\mathcal{B}$ overestimated the actual average. We see that $\mathcal{B}$ gave a very large overestimate when the bounty rate was one and was close to correct when the bounty rate was $\leq 10^{-4}$.

Discontiguous Tasks This experiment illustrated the role that the bounty rate plays in an environment where the nearest neighbor approach struggles.

For this experiment the bounty rate was set to five. We created two regions of 40×40 that were centered at (20,20) and (150,150) as illustrated in Figure 3. There were two depots at the centers of the two regions and the agent started in the region centered at (20,20). Tasks were generated with the same ρ values as above, but each region had a mean task generation rate of $\lambda_1 = \lambda_2 = \frac{1}{32}$ which due to the property of Poisson processes had an overall generation rate $\lambda = \frac{1}{16}$. This experiment was run for 1,000,000 time steps rather than 300,000 due to the larger area in which the agents operate.

Results As we see in Table 2 the nearest neighbor approach performed very poorly compared to the bounty hunting approach. This was because the bounty rate motivated the agents to more frequently service tasks that were further away.

6.2 Four Agents

Next we considered two different four-agent cases where we adjusted the values for λ and $\bar{s}$ to see whether bounty hunting could match and improve on the NN approach with equitable partitions.

Rapid Task Generation Here the environment was an 80×80 square, and in order to maintain the same value for ρ we set $\lambda = \frac{1}{4}$ and used the same values for $\bar{s}$. The depots were located at (20, 20), (60, 20), (20, 60), and (60, 60) and each agent started at a different depot. The space was split into four, 40×40 square regions with the depot at the center of the region. Each nearest neighbor agent was assigned to service the tasks that were generated within its assigned region.

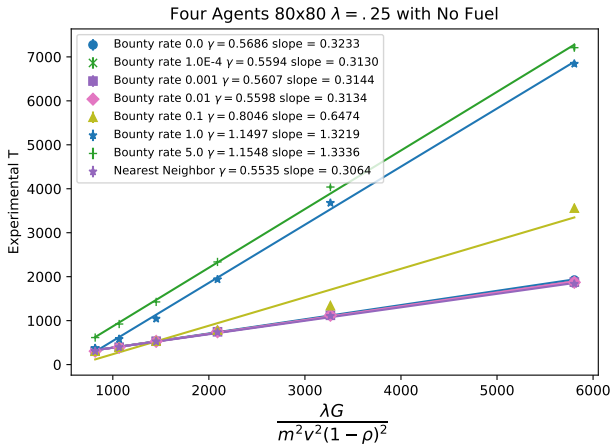

Figure 4: Rapid Task Generation Experiment: Four Agent, $\lambda = \frac{1}{4}$, 80×80 area with no fuel, 300,000 steps.

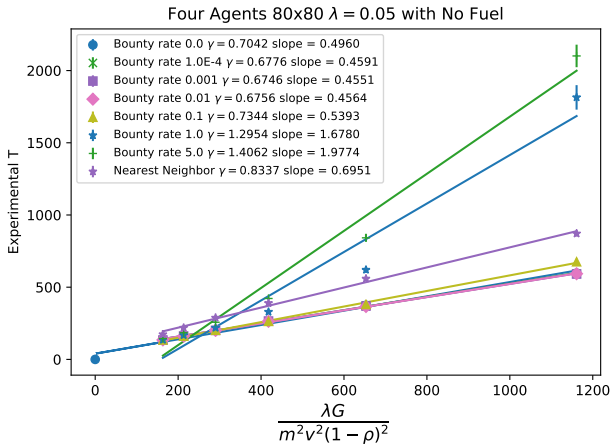

Figure 5: Slow Task Generation Experiment: Four Agent, $\lambda = \frac{1}{20}$, 80×80 area with no fuel, 300,000 steps.

This experiment focused on whether explicitly splitting the space into equitable partitions is necessary or whether a different approach could produce similar results.

Results In Figure 4 we plot the experimentally obtained waiting time in order to acquire the value of γ for each setting. We see that the bounty hunting approach did similarly to the nearest neighbor approach.

Slow Task Generation Again we considered an 80×80 environment with four agents, but we set $\lambda = \frac{1}{20}$ and we set the mean service times to $\bar{s} = 65, 60, 55, 50, 45, 40$. Therefore, we were still using the same value for ρ, but with different values for λ and $\bar{s}$. The agents and depots were all located in the same manner as in the previous experiment.

In this experiment, we examined a scenario in which explicitly splitting up the space performed poorly, to see whether our alternative approach could improve performance.

Results As we can see in Figure 5 the value for γ for the nearest neighbor approach was significantly greater than that of the bounty hunting approach with a bounty rate less than one. This meant that we could achieve superior results to the NN with equitable partitions.

6.3 Sixty Four Agents

Based on Equation (1), by equitably partitioning the space the system will scale as the number of agents increases. However, because we did not theoretically prove that the agents were equitably partitioning the space we demonstrated empirically that the average waiting time continued to be similar to or outperformed equitable partitions. The following two experiments sought to do this in addition to examining the impact of fuel.

In order for the system to scale we had to limit the communication between the agents. Specifically, the bounty hunters could only communicate with other agents within a radius of 40 units, and were aware of tasks within a radius of 40.

Without Fuel We made an 8×8 grid of evenly spaced depots (each depot centered in a 40×40 area). For this experiment we increased the size of the environment to a 320×320 area and we set $\lambda = \frac{64}{80} = 0.8$ and $\bar{s} = 65, 60, 55, 50, 45, 40$. Therefore, we maintained the same load factor values as in previous experiments: $\rho = 0.8125, 0.75, 0.6875, 0.625, 0.5625, 0.5$.

Results We see in Figure 6 that not only did the system scale, but the value of γ was dependent on the values for λ and $\bar{s}$. As expected, the bounty hunting approach outperformed the NN approach as we have a similar setting to that in Figure 5.

With Fuel We performed the same experiment except the agents had a fuel capacity of 3000 and a starting bounty balance of 1000.

Results The case where the agents were given unlimited fuel capacity and the case where the agents had a fuel capacity of 3000 units produced similar results. Even with a large number of agents, as long as we had enough depots and the agents did not need to make frequent trips to the depot to refuel, the system time was not degraded compared to NN.

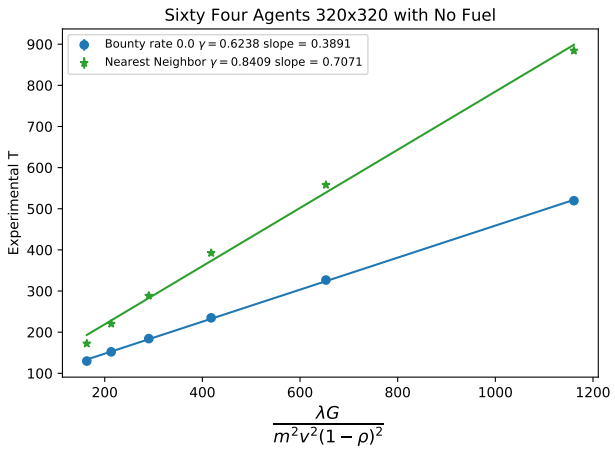

Figure 6: Sixty Four Agent Without Fuel Experiment: 320×320 area, samples taken after 300,000 steps.

7 Conclusions and Future Work

In bounty hunting we proposed a multiagent systems approach to solving the dynamic traveling repairman problem. Bounty hunting addresses a series of failures in current approaches, largely due to the unique flexibility of the bounty rate. It performs efficiently in a discontiguous environment thanks to the ability of the bounty rate to bring distant tasks "near." The bounty rate can, additionally, improve bias and variance of the system, adding dimensions of efficiency not studied by current approaches.

Furthermore, the ability of bounty hunting agents to learn not to go after tasks other agents have signaled enables bounty hunting to not only match but in some cases outperform the NN equitable partitions approach, without explicitly splitting up the space. Therefore, the bounty hunting approach Pareto dominates the NN approach. Also, this flexibility scales well. Bounty hunting offers an effective alternative to equitable partitions in heavy load settings.

For future work we hope to prove theoretically that the bounty hunters are splitting up the space equitably in an online fashion, and to explore bounty hunting's application to other vehicle routing problems, such as pick-up and delivery.

References

[1] Dimitris Bertsimas and Garrett van Ryzin. The dynamic traveling repairman problem. 1989.

[2] Dimitris Bertsimas and Garrett van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, 39(4):601–615, 1991.

[3] Dimitris Bertsimas and Garrett van Ryzin. Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles. *Operations Research*, 41(1):60–76, 1993.

[4] Dimitris Bertsimas and Garrett van Ryzin. Stochastic and dynamic vehicle routing with general demand and interarrival time distributions. *Advances in Applied Probability*, 25(4):947–978, 1993.

[5] Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *ACM symposium on Theory of computing*, pages 163–171, 1994.

[6] B. P. Gerkey and M. J. Matarić. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[7] Per Brinch Hansen. An analysis of response ratio scheduling. In *IFIP Congress (1)*, pages 479–484, 1971.

[8] Jiangchuan Huang and Raja Sengupta. System time distribution of dynamic traveling repairman problem under the part-n-tsp policy. In *European Control Conference*, pages 2762–2767, 2015.

[9] Michael K. Hui and Lianxi Zhou. How does waiting duration information influence customers' reactions to waiting for services? *Journal of Applied Social Psychology*, 26(19):1702–1717, 1996.

[10] Sandy Irani, Xiangwen Lu, and Amelia Regan. Online algorithms for the dynamic traveling repair problem. *Journal of Scheduling*, 7(3):243, 2004.

[11] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton Kleywegt, Sven Koenig, Craig Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *In Robotics: Science and Systems*, 2005.

[12] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[13] Marco Pavone, Alessandro Arsie, Emilio Frazzoli, and Francesco Bullo. Equitable partitioning policies for robotic networks. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2356–2361. IEEE, 2009.

[14] Marco Pavone, Emilio Frazzoli, and Francesco Bullo. Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment. *IEEE Transactions on Automatic Control*, 56(6):1259–1274, 2011.

[15] GA Shidali, SB Junaidu, and SE Abdullahi. A new hybrid process scheduling algorithm (pre-emptive modified highest response ratio next). *Computer Science and Engineering*, 5(1):1–7, 2015.

[16] Drew Wicke, David Freelan, and Sean Luke. Bounty hunters and multiagent task allocation. In *International Conference on Autonomous Agents and Multiagent Systems*, 2015.

[17] Drew Wicke and Sean Luke. Bounty hunting and human-agent group task allocation. In *AAAI Fall Symposium on Human-Agent Groups: Studies, Algorithms and Challenges*, 2017.

[18] Drew Wicke, Ermo Wei, and Sean Luke. Throwing in the towel: Faithless bounty hunters as a task allocation mechanism. IJCAI workshop on Interactions with Mixed Agent Types, 2016.

[19] Adam Wierman. Fairness and classifications. *ACM SIGMETRICS Performance Evaluation Review*, 34(4):4–12, 2007.

[20] Haiping Xu. *Optimal policies for stochastic and dynamic vehicle routing problems*. PhD thesis, Massachusetts Institute of Technology, 1994.