Determining Effective Swarm Sizes for Multi-Job Type Missions*

Meghan Chandarana[1], Michael Lewis[2], Katia Sycara[3], and Sebastian Scherer[3]

***Abstract*— Swarm search and service (SSS) missions require large swarms to simultaneously search an area while servicing jobs as they are encountered. Jobs must be immediately serviced and can be one of several different job types – each requiring a different service time and number of vehicles to complete its service successfully. After jobs are serviced, vehicles are returned to the swarm and become available for reallocation. As part of SSS mission planning, human operators must determine the number of vehicles needed to achieve this balance. The complexities associated with balancing vehicle allocation to multiple as yet unknown tasks with returning vehicles makes this extremely difficult for humans. Previous work assumes that all system jobs are known ahead of time or that vehicles move independently of each other in a multi-agent framework. We present a dynamic vehicle routing (DVR) framework whose policies optimally allocate vehicles as jobs arrive. By incorporating time constraints into the DVR framework, an M/M/k/k queuing model can be used to evaluate overall steady state system performance for a given swarm size. Using these estimates, operators can rapidly compare system performance across different configurations, leading to more effective choices for swarm size. A sensitivity analysis is performed and its results are compared with the model, illustrating the appropriateness of our method to problems of plausible scale and complexity.**

I. Introduction

Many envisioned applications of robotic swarms require that jobs be immediately serviced by swarm members. In forest fire applications, a swarm may be tasked with searching a section of the forest during a wildfire for brush fires that have sparked due to embers carried by the wind. Those discovering a new fire must act immediately to put it out rather than waiting for assistance and risk losing the current wildfire containment level. Similarly, in military applications, a swarm on patrol may come across a Scud missile preparing to launch, where any action not taken immediately has little value. Less extreme examples, such as breaking small teams off to maintain surveillance of a newly detected suspicious site, follow a similar pattern. We call this type of application in which members of a searching swarm, upon detection of a job, are immediately dispatched to service it a Swarm Search and Service (SSS) mission. Once a job is serviced vehicles return to the swarm for reallocation elsewhere.

When planning SSS missions one of the main challenges human operators face is determining the number of vehicles needed to effectively handle the expected jobs (and their varying job types). Knowing only the expected numbers of jobs of each job type and the size of the search area, predicting the number of vehicles needed to handle a variety of load conditions is extremely difficult for humans. Mission planning analysis tools will need to be developed to guide operators in effectively selecting the required swarm size if robotic swarms are to be fielded for practical applications.

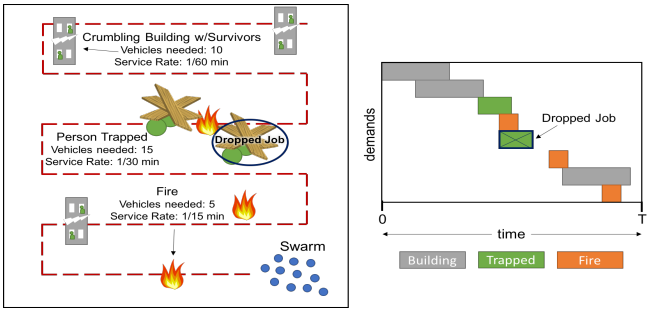

Fig. 1: Example SSS mission with three different job types (top) and timeline of each job's arrival time (bottom).

In SSS missions vehicles use local control laws (i.e., direct neighbor communication only) to search large areas and simultaneously service jobs. As the swarm moves with some predefined search pattern (red dashed line) jobs "arrive" as they enter the sensing range of a swarm member (Figure 1). Each job requires a vehicle or group of vehicles to break off from the main swarm for a specified amount of time to successfully service it. Once the job is serviced, vehicles return to the swarm and become available for reallocation. Multiple job types (e.g., fire, trapped people, etc.) with varying vehicle requirements may be present. For example, Figure 1 shows a swarm tasked with finding survivors after a natural disaster where survivors are both buried under fallen debris as well as inside crumbling buildings. In addition, any fires must be put out. Each rescue activity might require a different number of vehicles and service time. The schedule of each job's arrival is shown in Figure 1 (right). When too many jobs arrive at the same time, one or more may be dropped.

Research in robotic swarms typically concludes after a swarm has traversed a given area or each robot has moved to its assigned monitoring location but takes full credit for the subsequent activities that actually achieve the swarm's objective. The aim of the work presented in this paper is to develop a mission planning tool to assist human operators in determining effective swarm sizes required to successfully complete an SSS mission's objective. We adopt a dynamic vehicle routing (DVR) framework that leverages connected swarm communication networks to prescribe optimal policies

[1]The author is with Mechanical Engineering at Carnegie Mellon University, Pittsburgh, PA, USA {mchandar@cmu.edu}

[2]The author is with Information Sciences and Intelligent Systems at the University of Pittsburgh, Pittsburgh, PA, USA {ml@sis.pitt.edu}

[3]The authors are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, USA {katia@cs.cmu.edu, basti@andrew.cmu.edu}

*Sponsored by NASA LaRC (NIA Activity 201020)

for routing vehicle(s) to service dynamically arising jobs. We then model the SSS system as a variant of the DVR problem with time constraints presented by Bullo et al. in [1], which aims to determine the minimum number of vehicles needed to service the jobs of all the job types in the system at a prescribed steady state level of success. This problem adds the consideration of a patience time – the amount of time after a job appears that it can wait to be serviced before it is dropped. Jobs that are not serviced cannot re-enter the queue. In SSS missions routing is solved by setting patience time to the time for a vehicle to travel to the edge of its sensing radius, thus dropping any job that cannot be serviced immediately. An M/M/k/k queuing method is presented for determining the number of vehicles needed to achieve a prescribed level of success, where we equate the probability of successfully servicing jobs within patience time to its complement, the probability of dropping a job. An SSS mission is simulated and a numerical sensitivity analysis is presented.

The contributions of this work are: (1) optimal swarm vehicle routing policies for SSS missions, (2) treatment of time between jobs sensed by a searching swarm as an arrival rate for a queuing model and (3) the development of an M/M/k/k model for the prediction of optimal swarm size adapting results from CPU resource allocation literature.

The remainder of the paper is organized as follows. Section II reviews current work on vehicle routing problems. Section III outlines a vehicle routing framework for SSS missions and introduces optimal policies for selecting service vehicle(s). Section IV presents an M/M/k/k model for solving the DVR problem with time constraints applied to an SSS mission. Section V compares the performance of a simulated swarm system with that of the M/M/k/k model. A discussion of how the presented method can be used to plan swarm missions is given in Section VI. Lastly, Section VII provides concluding remarks and directions for future work.

II. Related Work

Past work has modeled the assignment of vehicles to jobs as a multi-agent static vehicle routing problem (SVR) where all system jobs are fixed and known ahead of time. Vehicles service jobs by visiting job locations. No new jobs appear and the locations of all the jobs are known [2], [3]. This work aims to determine the assignment and schedule of servicing jobs while minimizing cost. Centralized solutions that utilize a multiple travelling salesmen problem formulation have been presented [4]. Moore and Passino provide a distributed method solving the mobile agent task allocation problem [5].

Unlike the multi-agent SVR problem where we know all jobs and their locations ahead of time, a dynamic vehicle routing (DVR) framework assumes that jobs arrive during mission execution [1] and are known to all agents. Rather than finding routes as in SVR, DVR solutions involve finding policies for selecting the best vehicle(s) to service an incoming job to achieve desired objectives such as minimizing waiting times and travel distance [1]. Bertsimas and Van Ryzin introduced queuing methods to solve DVR problems where vehicles move in straight lines to visit jobs whose locations and arrivals are stochastic [6], [7], [8]. By analyzing a DVR problem from an algorithmic queuing theory perspective, traditional queuing techniques have been leveraged to develop effective policies that account for system-level constraints to optimize system performance in a general steady state case rather than tailoring performance to a single set of system service demands. Various constraints have been studied: time constraints [9], [10], service priorities [11], vehicle dynamics [12], [13], limited sensing range [14] and team formation [15]. This paper uses time constraints to model the necessary number of vehicles required in SSS missions to achieve a given performance (Section IV)

III. DVR Framework for Swarms

The SSS problem can be framed as a variant of the DVR problem. Unlike the work presented in [1], jobs are identified as they are sensed by a swarm member as opposed to being sensed at the time they appear in the environment (whether vehicles can sense them or not) by an omniscient observer and relayed to the vehicles.

Therefore, the arrival rate of jobs in an SSS mission corresponds to the time required for a swarm to travel within sensing range of the next job. Since new jobs arrive as vehicles sense them, the steady state performance can be analyzed using algorithmic queuing theory. Let $\pi \in \mathcal{S}$ be a stable routing policy, T_π be the system time of a policy, D_π be the total distance traveled for system, and C_π be the total system cost for a given policy. The SSS problem can then be defined as finding an optimal policy $\pi^* \in \mathcal{S}$ such that

$$C^* := C_{\pi^*} = \inf_{\pi \in \mathcal{S}} C_\pi \tag{1}$$

where C^* is the optimal system performance cost.

$$\begin{aligned} C_{\pi^*}(T_{\pi^*}, D_{\pi^*}) &= \min_{\pi}(T_\pi + D_\pi) \\ &= \min_{\pi} \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} t_\pi(j, \alpha) + d_\pi(i) \end{aligned} \tag{2}$$

where α is the job type, $\mathcal{I}$ is the set of all robots in the swarm, $t_\pi(j, \alpha)$ is the time to service each job j of type α in the set $\mathcal{J}$, and $d_\pi(i)$ is the distance each robot i travels. The optimal policy π^* is one that minimizes the total system time required to service each job as it arrives and minimizes the total distance traveled by the servicing vehicle(s).

We assume a uniform spatial density function in which jobs are randomly and uniformly distributed. A non-uniform spatial density would arise if some regions were more populated or job locations were imprecisely known ahead of time. We will not consider this more complex situation at this time, although the base case we address provides a lower bound on performance [1].

Lemma 1: Jobs that are randomly and uniformly distributed will approximate a Poisson arrival rate.

Proof: Let us assume that our environment can be decomposed into a uniform grid where the probability of a job being present in any given grid cell is uniform, equal

and given by a binomial distribution $P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$. If $n\to\infty$ and $p\to 0$, then $\lambda=np$ remains constant [16] and

$$\begin{aligned}\lim_{n\to\infty} P(X=k) &= \lim_{n\to\infty}\binom{n}{k}p^k(1-p)^{n-k}\\ &= \lim_{n\to\infty}\frac{n!}{k!(n-k)!}\binom{\lambda}{n}^k\left(1-\frac{\lambda}{n}\right)^{n-k}\\ &= \frac{\lambda^k e^{-\lambda}}{k!}\end{aligned} \tag{3}$$

Therefore, the limit reaches a Poisson distribution. The commonly used rule of thumb for good approximation of large n (>20), number of jobs, and small p (<0.05), rate at which jobs are dropped, is met by our application. ■

If we assume that the job detecting vehicle is always the nearest vehicle, the SSS policy can be prescribed according to the geometric relationship between the new job and the vehicle. By leveraging swarm communication networks the swarm can distinguish between free and allocated vehicles. We prove that assigning the job detecting vehicle (or next nearest vehicle, if we stipulate the detecting vehicle cannot be assigned) to service the new job is an optimal policy with respect to wait time and travel distance.

Consider m vehicles moving at speed v within $\mathbb{R}^2$. Jobs arrive within a bounded convex set $\mathcal{H}$ according to a Poisson process with arrival rate λ_α, where α corresponds to a specific job type. Their locations are independent and identically distributed (i.i.d.) according to a density whose support is $\mathcal{H}$. A job's location is only known (sensed) at its arrival time. At each job location, we assume that vehicle(s) spend a given amount of time so that the job can be completed. After vehicle(s) have completed the service the job is removed from the queue and vehicles return to the swarm.

Let us assume that vehicles in our swarm are in one of two modes: navigation/search mode or job servicing mode. Only vehicles that are in navigation/search mode (i.e., not allocated to a job yet and are still part of the main swarm) are able to sense new jobs that need to be serviced. In situations where jobs only require one vehicle to service them, there are no conflicts and all vehicles are homogeneous, we define the following policy:

Closest Vehicle Policy – The vehicle that sensed the new job is assigned to service the job.

Lemma 2: The **Closest Vehicle Policy** is the optimal policy in all load conditions.

Proof: Since the sensing robot is the closest to the job when it arrives, $d_s < d_i$ for $i \neq s, i \in \{1,...,m\}, s \in \{1,...,m\}$ where d_s is the distance between the sensing robot and the job and d_i is the distance between another robot in the swarm and the job. The time required to finish servicing a job can be written as $t_\alpha = t_r + \mu_\alpha$, where t_r is the time required for the vehicle to reach the job and μ_α is the time required for a vehicle to spend at the given job location to complete the job. μ_α is the same for all vehicles. The time required to travel to the job location can be written as $\frac{d_s}{v}$ and $\frac{d_i}{v}$ for the sensing vehicle and all other vehicles respectively. Therefore, the vehicle that senses the job (and therefore closest to it) will be able to service the job the quickest. If cost is scaled based on the distance a vehicle is required to travel to complete the service, the sensing vehicle also performs the service with the lowest cost. ■

The **Closest Vehicle Policy** implies that vehicles that are left within the swarm are the ones that should be assigned to service new jobs as they arrive. If the sensing vehicle is chosen to service the job, both the time to service the job and the cost are minimized.

In cases where a vehicle in the swarm has conflicts (i.e., senses two jobs simultaneously), then the following policy can be defined:

Next Closest Neighbor Vehicle Policy – A vehicle senses more than one job simultaneously. The sensing vehicle services the closest job, or picks one randomly if they are both equal distance away. For each additional job that needs to be serviced, the sensing vehicle then asks their neighbor closest to the job to perform the service.

In the case of a job that requires multiple vehicles simultaneously to service it successfully, the **Closest Group Policy** can be defined:

Closest Group Policy – A vehicle senses a new job that requires multiple vehicles to service it simultaneously. The sensing vehicle assigns itself as group leader. If more than one job is sensed, the sensing vehicle chooses the closest job to service (or picks one randomly in the case of equidistant jobs). It assigns itself as group leader for that chosen job. For each remaining job, the sensing vehicle asks a free neighbor that is closest to the job to be the leader. Each group leader then gathers enough neighbors to service the job. If the vehicle has fewer neighbors than the required number, its neighbors ask their neighbors until enough vehicles have been assigned.

Similar to what is shown above, the **Next Closest Neighbor Vehicle Policy** and the **Closest Group Policy** will service the job in the minimum time with the lowest cost.

IV. DVR WITH TIME CONSTRAINTS

We additionally consider the problem of DVR with time constraints. In this context we exclude priorities and vehicle motion constraints. As defined by Bullo et al. in [1], the aim is to: *Find the minimum number of vehicles needed to ensure that the steady-state probability that a job is successfully serviced is larger than a desired value*. This problem would be seen in swarm mission planning where human operators are tasked with determining the number of vehicles to deploy given expected numbers of various job types. This is represented by the following:

$$\min_{\pi}|\pi|, \text{subject to } \lim_{\alpha\to\infty} P_\pi[W_\alpha < G_\alpha] \geq \phi_d \tag{4}$$

where α is a system job type, W_α is the wait time of job type α, G_α is the given accepted patience time (amount of time a job can wait to be serviced) of job type α and ϕ_d is the threshold for system performance. In general, patience times are job type dependent, but in SSS we assume they are all the

same and equal to the time for a vehicle to travel to the edge of its sensing radius (i.e., jobs are serviced immediately).

Within the DVR framework, since many SSS missions require new jobs to be immediately serviced, the steady state system performance can be modeled as an infinite horizon M/M/k/k queue system where jobs enter the queue as they come within sensing range of a swarm member. The length of the queue is dependent on whether there are enough vehicles to service the new job at its time of arrival. If there are not enough vehicles, the job is dropped. We equate the probability of a job being serviced within its accepted patience time (G_α = time to travel to a vehicle's sensing radius) to the probability of a job being dropped:

$$\min_{\pi} |\pi|, \text{subject to} \lim_{\alpha \to \infty} R_\pi \leq \delta \tag{5}$$

where R_π represents the probability of the system dropping a job upon its arrival due to lack of available resources and δ is the accepted drop rate.

A. M/M/k/k Prediction Model

In an M/M/k/k queuing system jobs arrive according to a Poisson arrival rate and service times are exponentially distributed. There are k servers (or N vehicles in the SSS mission context) in the system that are able to service incoming jobs. Unlike more traditional M/M/k systems where queues have an infinite size, M/M/k/k systems have a limited size queue equal to the number of total servers in the system. Using the M/M/k/k framework the SSS system is one with k servers where arriving jobs are parallel (i.e., they require the use of multiple servers simultaneously). Jobs arrive according to a Poisson process with rate λ_α, where α is the job type. Type α jobs require i_α servers simultaneously (i.e., i_α vehicles). Each job type is serviced for $Exp(\mu_\alpha)$ time, where μ_α is the service rate. After the job is completed, all used servers are free once again.

Within computer applications M/M/k/k systems are used to model multiple users competing for a limited number of shared resources. Let $s = (n_1, ..., n_k)$ be the state of the system where n_α is the number of jobs in the system of type α. The probability of a system being in a particular state (i.e., probability that a particular number of each job type are present in the system) can be defined [17], [18]:

$$P_\pi(s) = \prod_{\alpha=1}^{k} \frac{\rho_\alpha^{n_\alpha}}{n_\alpha!} \cdot C \tag{6}$$

$$C = \left(\sum_{s \in \mathcal{S}} \prod_{\alpha=1}^{k} \frac{\rho_\alpha^{n_\alpha}}{n_\alpha!} \right)^{-1} \tag{7}$$

$$\rho_\alpha = \frac{\lambda_\alpha}{\mu_\alpha} \tag{8}$$

where $P_\pi(s)$ is the probability the system is in a given state s, C is a normalizing constant, and ρ_α is the utilization factor. $\mathcal{S}$ is the set of all possible system states. The set of states, $\mathcal{S}$, is composed of states where various combinations of job types lead to the system being as fully utilized as possible. Since jobs in our queue are those that are sensed as the swarm searches the given area, the limited number of resource (or servers) are the vehicles in our swarm. If no jobs of type α exist in our system, all states with $n_\alpha > 0$ are ignored and not included in $\mathcal{S}$. Using the probability of the system being in a certain fully utilized state s, we can determine the probability that the system will not have enough vehicles to service a new job that arrives, resulting in a dropped job.

The rate of dropping a job for each state is calculated by

$$r_\pi(s) = P_\pi(s) \cdot \sum_{d \in \mathcal{D}} \lambda_d(s). \tag{9}$$

where $\mathcal{D}$ is the set of job types for which the system will be forced to drop the job given its current system state, s. λ_d is the arrival rate of a job type that would cause the system to drop the job given the current system state, s. The total drop rate for a given swarm size can be calculated as follows:

$$R_\pi = \sum_{s \in \mathcal{S}} r_\pi(s). \tag{10}$$

B. Grid World Example

For the remainder of the paper we will use a grid world example. In this example, a swarm travels at a constant velocity through a 100 x 100 grid. For convenience we will express time in terms of grid cells the swarm has traversed. The swarm traverses the cells in a boustrophedon (i.e., lawn mower) pattern. If a job is located at the cell the swarm has reached, the required number of vehicles to service it will be allocated and designated as "busy." They return to the swarm and are designated as "free" after a given number of cells have been traversed. If not enough vehicles are available to service the new job, then the job is dropped. Jobs are spread across the grid randomly from a uniform spatial distribution.

TABLE I: Grid World Job Service Rates

Job Type	Required No. of Veh.	μ_α (jobs/cell)
1	15	1/2500
2	5	1/5000
3	10	1/3000

TABLE II: Grid World Configurations

Configuration	Type 1 Veh.	Type 2 Veh.	Type 3 Veh.
1	5	5	5
2	5	10	15

We assume there are three different job types. Type 1 requires 15 vehicles for servicing, type 2 and 3 require 5 and 10 vehicles respectively. The service rates for each job type, μ_α, are shown in Table I. Two job configurations will be explored (Table II). In Configuration 1 there are 5 jobs of each job type. Configuration 2 has 5 jobs of type 1, 10 jobs of type 2 and 15 jobs of type 3 present. The remainder of this section will use the M/M/k/k model to determine the minimum required swarm size to achieve a desired drop rate in the context of the grid world example.

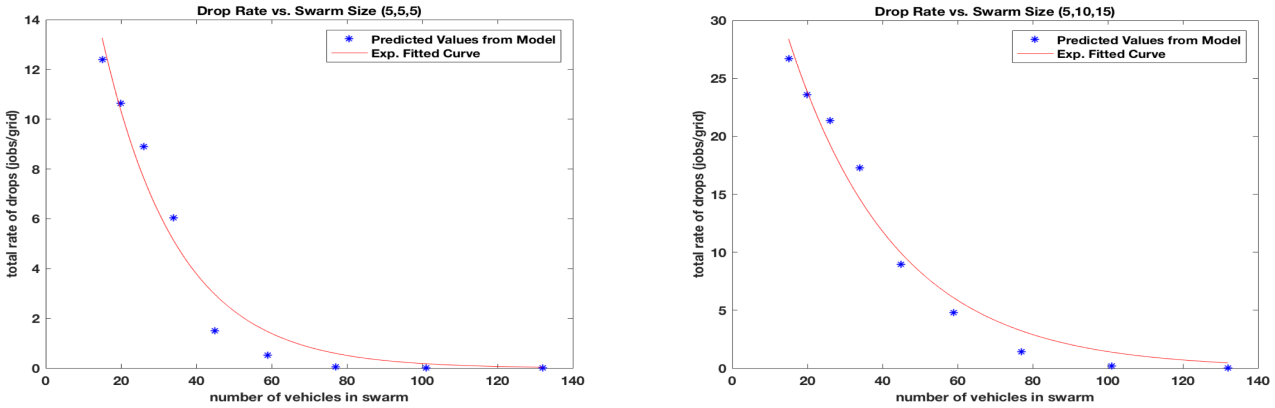

Fig. 2: Predicted drop rate versus swarm size.

TABLE III: Arrival Rates for Each Configuration

Configuration	Λ_1 (jobs/cell)	Λ_2 (jobs/cell)	Λ_3 (jobs/cell)
1	0.0005	0.0005	0.0005
2	0.0005	0.0010	0.0015

1) M/M/k/k Model Results: The arrival rate Λ_α of each of the three job types, α, in the grid world is assumed to be the expected value of that job type over the grid:

$$\Lambda_\alpha = E[\lambda_\alpha] = \frac{n_\alpha}{|\mathcal{G}|} \tag{11}$$

where n_α is the expected number of jobs that will be seen of job type α over the grid and $|\mathcal{G}|$ is the total number of cells in the grid (10,000 in this example). The arrival rates for both, Configuration 1 and 2 are shown in Table III. All values are expressed in jobs/cell traversed .

Using the job type parameters and their associated arrival rates shown in Table III, both configurations were run with the M/M/k/k model. The results are shown in Figure 2. For both configurations an exponential curve (red line) captures the data (blue *) well. The goodness of fit measures are shown in Table IV. Overall, Configuration 1 – where the number of expected jobs of each job type is the same – results in a lower drop rate (Figure 2 (left)). Each exponential curve can be used to estimate the dropped job rate for a particular swarm size given expected arrival rates for each job type.

TABLE IV: Goodness of Fit Measures

Measure	Configuration 1	Configuration 2
SSE	6.5673	21.9382
R^2	0.9682	0.9761
Adjusted R^2	0.9637	0.9727
RMS	0.9686	1.7703

V. Simulation

A grid world SSS mission was simulated in MATLAB. The three job types shown in Table I were used. Both configurations (Table II) were simulated. In each simulated mission the location of jobs was randomly distributed from a uniform spatial distribution. No two job types were allowed to occupy the same grid cell. Service rates for each job type were specified as either: 1) a fixed rate (expected arrival rates shown in Table I) or 2) a sampled value from an exponential distribution with mean λ_α. Each configuration was run with a swarm size of 30, 50 and 70. For each swarm size, 500 different missions were run.

The values shown in Figure 3 give the average mission drop rate across the 500 missions found from simulating both configurations. A fixed service rate and a service rate sampled from an exponential distribution were simulated for each configuration. In Figure 3, the mean drop rate (solid color line) is shown in terms of the entire size of the grid. The standard deviation is shown by the corresponding solid colored region. Fixed service rate results are shown in red, while exponential service rate results are shown in blue.

VI. Discussion

The results shown in Figure 3 indicate that in both configurations the M/M/k/k model is more effective at predicting the swarm's behavior if service rates are sampled from an exponential distribution as compared to being a fixed rate, however this leads to larger standard deviations. This can be attributed to the fact that the model was built assuming exponentially distributed service rates. However, the model still does a fairly good job of predicting performance in Configuration 1 given a fixed service rate. The results demonstrate the appropriateness of our model for predicting the steady state performance of job types with exponentially distributed service rates for missions of similar scale and complexity.

The sensitivity analysis in Section V highlights the trade-off seen between the dropped job rate and swarm size. If operators want to reduce the dropped job rate they must increase their swarm size exponentially. In Configuration 1 we see that increasing the swarm size from 30 to 70 vehicles decreases the drop rate from about 42% of the total number of jobs to about 6%. When the total number of jobs doubles in Configuration 2 (from 15 to 30) we see that the dropped job rate decreases from 56% to 14% as the swarm size increases from 30 to 70 vehicles. However, as the total

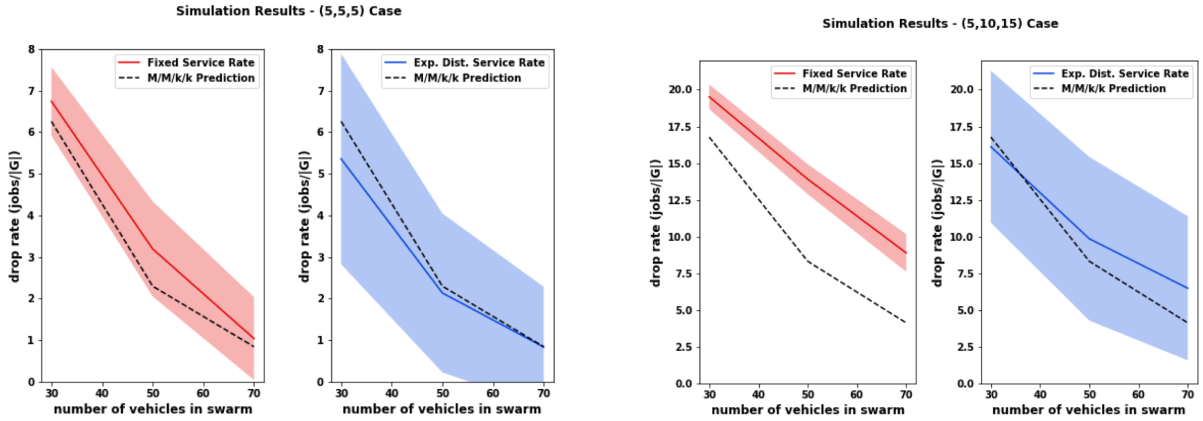

Fig. 3: Simulation results for both service rate specifications in each of the two configurations.

number of jobs (across all job types) increases, so does the overall number of dropped jobs.

By incorporating the developed model into mission planning tools and interfaces, human operators can quickly and easily compare system performance across different mission configurations. This will lead to a reduction in planning time, as well as an increase in the likelihood of an operator achieving the desired system performance. Lastly, by leveraging the presented predictive model, swarm mission success will be less dependent upon highly skilled operators, thereby making swarm systems more accessible to a broader user base.

VII. Conclusion and Future Work

The work presented in this paper explored a DVR framework for SSS missions. The DVR problem with time constraints was explored and an M/M/k/k solution was presented for determining the minimum number of vehicles needed in the swarm to achieve a desired steady state system performance. A sensitivity analysis in the context of a grid world example was conducted to compare the predicted results from the model to simulation results. Results indicate that the M/M/k/k model does a good job of predicting system performance with configurations containing equal arrival rates across job types and configurations with varying arrival rates. Both fixed service rates and those sampled from an exponential distribution were evaluated.

Future work will explore spatially biased jobs. This results in job types having different arrival rates depending on what region the swarm is in. As mentioned in [1] the system performance can be modeled by partitioning the environment into regions separated by the boundaries of distinct arrival rates. Each region would be analyzed individually. All results would be fused to provide an accurate prediction of the system steady state performance.

References

[1] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[2] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE transactions on robotics and automation*, vol. 18, no. 6, pp. 911–922, 2002.

[3] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous vehicle-target assignment: A game-theoretical formulation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 584–596, 2007.

[4] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[5] B. J. Moore and K. M. Passino, "Distributed task assignment for mobile agents," *IEEE Transactions on automatic control*, vol. 52, no. 4, pp. 749–753, 2007.

[6] D. J. Bertsimas and G. Van Ryzin, "A stochastic and dynamic vehicle routing problem in the euclidean plane," *Operations Research*, vol. 39, no. 4, pp. 601–615, 1991.

[7] ——, "Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles," *Operations Research*, vol. 41, no. 1, pp. 60–76, 1993.

[8] ——, "Stochastic and dynamic vehicle routing with general demand and interarrival time distributions," *Advances in Applied Probability*, vol. 25, no. 4, pp. 947–978, 1993.

[9] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler, "A stochastic and dynamic vehicle routing problem with time windows and customer impatience," *Mobile Networks and Applications*, vol. 14, no. 3, p. 350, 2009.

[10] M. Pavone and E. Frazzoli, "Dynamic vehicle routing with stochastic time constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1460–1467.

[11] S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli, "Dynamic vehicle routing with priority classes of stochastic demands," *SIAM Journal on Control and Optimization*, vol. 48, no. 5, pp. 3224–3245, 2010.

[12] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, 2008.

[13] J. J. Enright, K. Savla, E. Frazzoli, and F. Bullo, "Stochastic and dynamic routing problems for multiple uninhabited aerial vehicles," *Journal of guidance, control, and dynamics*, vol. 32, no. 4, pp. 1152–1166, 2009.

[14] J. Enright and E. Frazzoli, "Cooperative uav routing with limited sensor range," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006, p. 6208.

[15] S. L. Smith and F. Bullo, "The dynamic team forming problem: Throughput and delay for unbiased policies," *Systems & control letters*, vol. 58, no. 10-11, pp. 709–715, 2009.

[16] P. C. Consul and G. C. Jain, "A generalization of the poisson distribution," *Technometrics*, vol. 15, no. 4, pp. 791–799, 1973.

[17] E. Arthurs and J. Kaufman, "Sizing a message store subject to blocking criteria," in *Proceedings of the third international symposium on modelling and performance evaluation of computer systems: Performance of computer systems*. North-Holland Publishing Co., 1979, pp. 547–564.

[18] M. Harchol-Balter, *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.