From Honeybees to Internet Servers: Biomimicry for Distributed Management of Internet Hosting Centres

Sunil Nakrani[1]‡, **Craig Tovey**[2]

[1] Computing Laboratory, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

[2] School of Industrial and Systems Engineering, Georgia Institute of Technology, 765 Ferst Drive, NW, Atlanta, GA 30332-0205, USA.

E-mail: Sunil.Nakrani@gatech.edu

Abstract. A honeybee biomimetic algorithm for management of an internet hosting centre performs well in tests [1]. The biomimicry was inspired by the resemblance between the honeybee colony's problem of allocating foragers amongst flower patches to maximise nectar influx, and the host centre's problem of allocating servers amongst host customers to maximise revenue. This paper gives details of the two problems' many similarities, and explores the extent to which the homology holds. It concludes with a brief survey of other swarm intelligence applications, and some general speculations regarding their various degrees of success.

AMS classification scheme numbers: 68M14,68M20,90B22,90B50,92D50

Submitted to: *Journal of Bioinspiration & Biomimetics*

‡ Present address: School of Industrial and Systems Engineering, Georgia Institute of Technology, 765 Ferst Drive, NW, Atlanta, GA 30332-0205, USA.

1. Introduction

A honeybee imitative algorithm for management of an internet hosting centre performs well in tests [1]. The biomimicry was inspired by the resemblance between the honeybee colony's problem of allocating foragers amongst flower patches to maximise nectar influx, and the host centre's problem of allocating servers amongst host customers to maximise revenue. This paper gives details of the two problems' many similarities, and explores the extent to which the homology holds. It concludes with a brief survey of other swarm intelligence applications, and some general speculations regarding their various degrees of success.

2. Internet Host Centre Orchestration

2.1. Internet Host Centres

A burgeoning World Wide Web (WWW) user population depends upon the Internet for business-critical and day-to-day activities. Like long-established utilities such as the power grid or the telephony network, the Internet's computing infrastructure is required to provide performance guarantees such as high availability and fault tolerance [2]. Internet traffic variability creates unusually demanding scalability and availability requirements. Service request traffic often exhibits bursts above the mean for lengthy periods over a wide range of timescales, and varies in terms of peak-to-trough by factors of more than 100 in less than an hour [3]. As a consequence, an emergent model for Internet computing infrastructure entails centralised internet hosting centres composed of commodity server appliances, managing content delivery of multiple co-hosted services to global users for a hosting fee [4, 5, 6].

The internet hosting model has gained a significant foothold in the commercial world [5, 7] with combined revenue of major companies exceeding $30 billion in 2004. Hosting centres pool traffic variability, much as insurance companies pool risk. Servers are shared among co-hosted services thereby benefiting from an opportunity to dynamically provision server capacity to meet unpredictable demands whilst offering economy of scale benefits to service content owners [6]. Given that the hosting fee may be negotiated on the basis of requests served, maximisation of the total requests revenue becomes the prime incentive of the hosting centre.

2.2. Server Orchestration Problem

Server ensemble management, as depicted in Figure 1, involves orchestrating the servers in the ensemble to be time-varying compositional parts of the virtual servers (co-hosted services) in the hosting centre over a lengthy time horizon so as to maximise revenue collection from the hosting fee associated with each service. For the entire time horizon, orchestration decisions have to be made as to which servers in the ensemble are part of which virtual servers. At any moment, any server may be orchestrated to remain

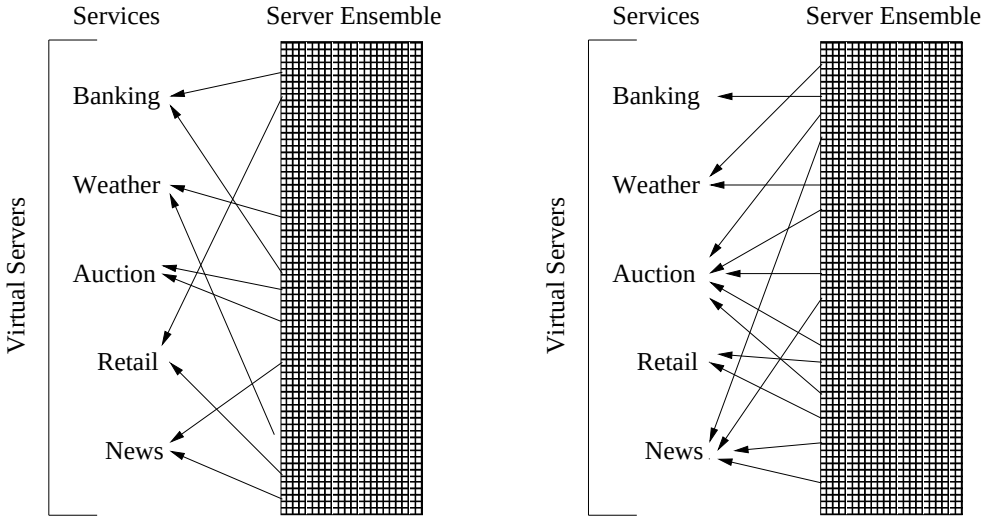

Figure 1. Server ensemble orchestration at two distinct time instants.

with its current given virtual server or to join another virtual server. If a server is orchestrated to leave a given virtual server and join another then the server in question suffers a downtime during which it earns no revenue. All these orchestration decisions have to be made in the face of unknown and highly variable future request arrivals for each of its co-hosted services. The average time to serve a request and fee paid per request may differ for each service along with request queueing behaviour in terms of how long a customer is willing to wait for a response. A revenue opportunity is lost if a user balks.

Suppose an internet hosting centre with N servers co-hosts M internet services. There are M virtual servers $V_1, \ldots, V_M$, hosting internet services with respective service request queues $Q_1, \ldots, Q_M$ buffering streams of arriving requests. Each virtual server V_i is composed of n_i servers, $n_i \geq 0$, such that $\sum_{i=1}^{M} n_i = N$. The unit of orchestration is a single server and the cost of orchestration is some fixed time units i.e. the server becomes unavailable for this fixed time interval as it undergoes the repurposing process. Discretize the time horizon into T time steps indexed by $t = 1, \ldots, T$ and let S_t denote the state of the system at the start of time step t, including the server membership of the virtual servers $V_1, \ldots, V_M$, status of any unavailable servers, and residual requests in the queues from time step $t-1$. Let $A_t = \{a_1^t, \ldots, a_M^t\}$ denote the request arrivals during time step t and $\pi^t = \{\pi_1^t, \ldots, \pi_M^t\}$ denote the server ensemble orchestration policy for time step t. Let $R(\pi^t, S, A)$ denote the revenue earned by an orchestration policy π^t during a time step t with initial state S and arrivals A, and similarly let $f(\pi^t, S, A)$ denote the state of the system which would eventuate at the start of the $(t+1)$st time step. Thus, the server orchestration problem is

$$\begin{aligned} &\textbf{Maximise:} && \textstyle\sum_{t=1}^{T} R(\pi^t, S_t, A_t) && \text{(total reward)} \\ &\textbf{Subject to:} && S_{t+1} = f(\pi^t, S_t, A_t) && \text{(Orchestration, Request Traffic)} \end{aligned}$$

and since the A_t are not known in advance, the problem is not deterministic.

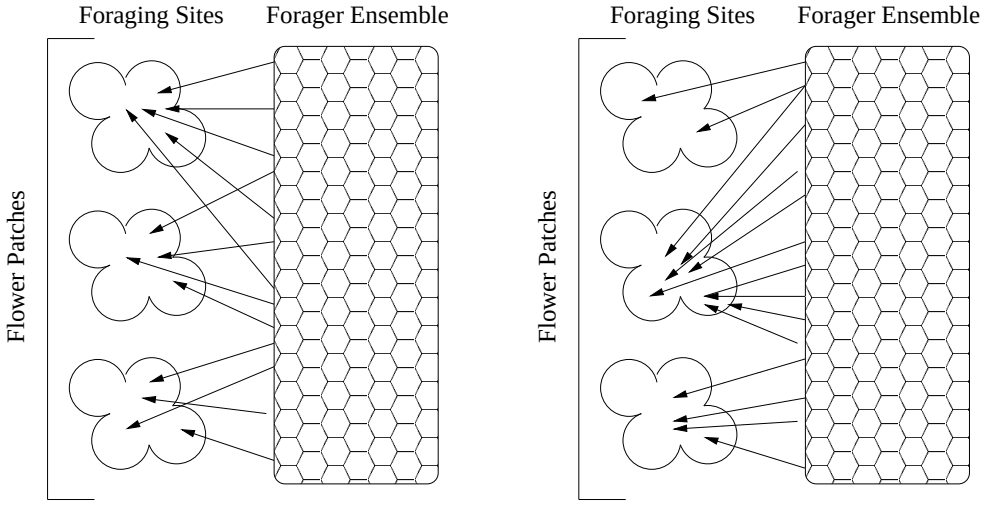

Figure 2. Foragers' orchestration at two distinct time instants.

3. Honeybee (*Apis mellifera*) Colonies: Forager Orchestration Problem

A typical honeybee colony comprises approximately 20-50 thousand bees with one queen, a few drones and the rest workers. Nectar collection is a principal foraging activity given that during summer the colony consumes approximately 70 kg of nectar, and approximately 50 kg of additional nectar is needed as a reserve for winter survival without which the queen freezes and the colony dies.

In colder climate zones such as Northern Europe, nectar is effectively available only in the summer for about 12 weeks each year. During this critical time period, a honeybee colony must orchestrate its foragers among the blooming flower patches in the surrounding countryside to collect the nectar it requires. However, the availability of nectar is unpredictable and the scale of fluctuation from dearth to abundance is unknown. From hour to hour and day to day, the availability and quality of nectar vary with micro-climatic conditions, sun position, blooming/withering cycle, and exploitation. It is not uncommon for a colony's nectar in-take rate to fluctuate by a factor of more than 100:1 within a day or so [8]. As illustrated in Figure 2, given the volatility of nectar supply, the colony must make time varying orchestration decisions as to the numbers of foragers to be deployed at each known flower patch. A forager bee can transfer from one flower patch to another, but the forager in question requires time to learn the location of the new flower patch during which she makes no contribution to colony's nectar collection endeavours.

Consider a honeybee colony with N nectar foragers and M flower patches. Let function $f_i(x_i)$ denote the value returned from the i^{th} flower patch with x_i foragers collecting nectar at the patch. Provided the value functions $f_i()$ do not change over

time then the resulting *static* forager orchestration problem is

$$\begin{array}{lll} \textbf{Maximise:} & \sum_{i=1}^{M} f_i(x_i) & \text{(Value function)} \\ \textbf{Subject to:} & x_i \geq 0 & \text{(Orchestration)} \\ & \sum_{i=1}^{M} x_i \leq N & \text{(Foragers).} \end{array}$$

Applying the Karush-Kuhn-Tucker conditions yields the optimal *static* solution $f_i^{'}(x_i) = \lambda \ \forall x_i > 0; f_i^{'}(0) \leq \lambda \ \forall x_i = 0$. That is, the marginal contributions are equal at each flower patch that is actively being foraged. This is intuitive since if the marginal contribution $f_i^{'}(x_i)$ at flower patch i exceeds the marginal contribution $f_j^{'}(x_j)$ at an active flower patch j then more nectar could be obtained by orchestrating a forager from flower patch j to i. The problem faced by the honeybee colony is more challenging than the static version outlined above, and is akin to the internet host problem described previously. Flower patch availabilities and profitabilities are highly variable, but responsiveness must be traded off against the temporary loss of the forager's productivity when she is recruited to a different patch.

4. Self-Organising Honeybee Forager Orchestration

Colonies of social Insect possess what has been classed as *Swarm Intelligence* [9]. A broad definition of the term implies a sophisticated collective behavior borne out of primitive interactions among members of the group to solve problems beyond the capability of individual members. Such colonies are characterised by (i) Self-Organisation: Decentralised and unsupervised coordination of activities, (ii) Adaptiveness: Response to dynamically varying environments and (iii) Robustness: Accomplishing the group's objective even if some members of the group are unsuccessful.

A model of self-organising behaviour that takes place within a colony of honeybees has been presented by Seeley [8]. The model describes interactions between members of the colony and the environment that leads to dynamic distribution of foragers to efficiently collect nectar from an array of flower patches that are capricious in terms of their value to the colony. The honeybee colony, as depicted in Figure 3, acquires information with respect to foraging prospects in the surrounding countryside continuously and acts upon this information in concert with nectar requirements of the colony to orchestrate foragers amongst prospective flower patches. The self-organising behaviour model comprises the following key elements:

Information Procurement: A honeybee colony procures two different types of information. It must not only continually refresh information with respect to flower patches currently being capitalised by the colony but also procure information with respect to any new enticing flower patches.

(i) Forager: A forager bee engaged in nectar collection at a flower patch returns to the beehive not only with nectar but also with information about the patch. Since a typical colony has several thousand bees foraging, the colony can procure and continually refresh information about current foraging assets.

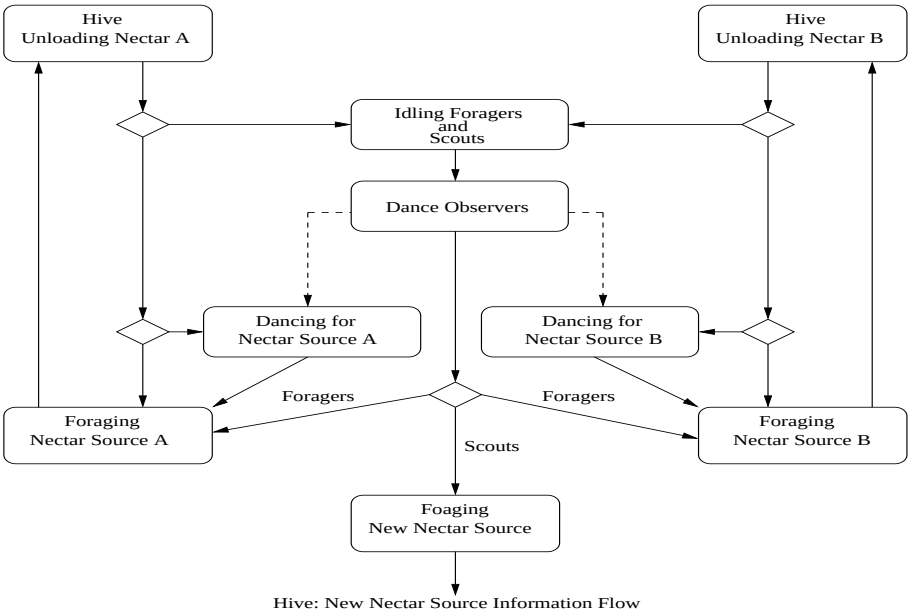

Figure 3. The information flow and decision processes in a honeybee colony (adapted from Seeley [8]).

(ii) Scout: Amongst several thousand foraging bees, a small percentage($\approx$10%) of bees are scouts. Scout bees behave differently from foragers in that scouts are engaged in unearthing new flower patches and, therefore, return to the beehive with information about possible new foraging assets that may prove to be profitable to the colony.

Information Type: On each visit to a flower patch, a forager bee returns to the hive with two types of information along with nectar:

(i) Value: During the process of nectar collection at a flower patch, a forager bee utilises her local wisdom about the patch defined by variables such as nectar sugar concentration, nectar bounty and distance from the hive and an internal scale of quality to determine patch value rating.

(ii) Location: In addition to value rating, a forager also has information about the geographical location of the flower patch in relation to the position of the sun and the beehive.

Information Filtering: Given the continual influx of a large volume of information to the beehive about newly discovered and known flower patches, the colony ensures that information pertaining to high value rated patches is broadcast whilst discarding information about relative low value rated patches. This has an implication that the information broadcast is restricted to the most valued patches available to the colony:

(i) Foraging Status: Each foraging bee returning to the hive must find a receiver bee to offload collected nectar. It is the job of receiver bees to store nectar in the beehive and the time it takes to find a receiver bee provides a cue as to

the nectar influx rate of the colony.

(ii) Filtering Threshold: Based on nectar influx rate cue as well as variables such as current climatic conditions and time of the day, forager bees adapt their sensory-threshold which decides whether information pertaining to her patch is to be broadcast or suppressed. Generally, the threshold is higher when the nectar influx rate is high and vice-versa.

The filtering mechanism enables a colony to capitalise on the most valued patches when the supply of nectar is abundant and be selective as to where foraging effort should be concentrated whilst capitalising on a wide range of low valued patches during periods of nectar scarcity.

Information Broadcast: Each forager, having delivered the nectar payload on her return to the beehive and, in the process, set its filtering threshold, will decide whether to broadcast information pertaining to her patch or not. If the threshold is crossed then she will broadcast the information using the communication signal and medium at her disposal:

(i) Signal Mechanism: To disseminate information about her patch, foragers use a signal mechanism known as waggle dance.

(ii) Signal Medium: Forager bees use an area located inside the beehive near the entrance known as the dance floor to convey her patch information.

Information Emphasis: If the criteria for dancing are satisfied, forager bees combine the decision to waggle dance with a value rating to present the information that conveys the degree of goodness and location of her patch:

(i) Dance Potency: A forager bee expresses the goodness of her patch via a series of waggle runs that constitute the waggle dance. If a patch is deemed to be highly valuable then it will elicit a waggle dance comprising of greater number of waggle runs. Effectively, the value of a flower patch is represented by dance duration.

(ii) Patch Location: A forager bee expresses her flower patch location by the direction of waggle runs on the dance floor. If the patch is located directly in line with sun then this is expressed by waggle run in straight vertical direction. Any angular deviation from this vertical represents an angle between the sun (azimuth) and a flight direction from the beehive to the patch.

Information Audience: Given the continual bombardment of information on a colony pertaining to known and newly discovered flower patches, there are some foragers that are consumers whilst others are oblivious to this information.

(i) Oblivious: There are two groups of foragers:

(a) This is a group of active foragers that are currently busy collecting nectar from flower patches. Each member of the group has a local knowledge of their own patch and does not use the global information available.

(b) This is a group of scout foragers that do not use the global information available. Instead, they tend to explore independently for new flower patches.

(ii) Consumer: This is a group of idling foragers that use the available global information and are potential candidates for deployment to flower patches.

Information Utilisation: The behaviour of foragers known as consumers is driven by the process of sampling and deciphering the global information being presented on the dance floor:

(i) Dance Sampling: Even though an array of waggle dances are being performed on the dance floor, a forager does not take the opportunity to review and select the dance that represents the most valuable patch but, instead, randomly select a dance.

(ii) Forager Response: Having selected a dance at random, it follows the dance carefully for several waggle runs to learn the location of the patch and leave the hive to collect nectar. There may be several attempts before it successfully finds a patch with each failure resulting in return to the hive and following a randomly selected dance.

Fundamentally, the self-organising forager orchestration in a honeybee colony emerges due to information flow into the colony and its response to this information. In summary, foraging bees visiting flower patches return to the hive with nectar and with a value rating (function of nectar quality, nectar bounty and distance from the hive) of respective flower patches. At the hive, forager bees interact with receiver bees to offload collected nectar. This interaction provides feedback on the current status of nectar flow into the hive. This feedback mechanism sets a response threshold for an enlisting signal. An amalgamation of response threshold and value rating influences the length of the enlisting signal known as the *waggle dance*. The waggle dance is performed on the dance floor where inactive foragers can observe and follow. Effectively, each active forager bee provides feedback on her local flower patch while observing bees have access to the set of attractive food sources being capitalised by the colony. However, individual foragers do not acquire the full set of global knowledge but rather randomly select a dance to observe from which they can learn the location of the flower patch and leave the hive to forage. The self-organising proportionate orchestration pattern, derived from multiple and proportionate feedback on goodness of food sources, is described by Seeley *et al.* [10] and validated by experimental study on a natural honeybee colony. The model given by Bartholdi *et al.* [11] predicts a steady state pattern of forager orchestration where rate of value accumulation equalises among forage sites being exploited i.e. self-organising orchestration results in equal average nectar return $\frac{f_i(x_i)}{x_i} = \mu \; \forall i$ where $f_i(x_i)$ denotes the value returned from x_i foragers collecting nectar at the i^{th} forage site.

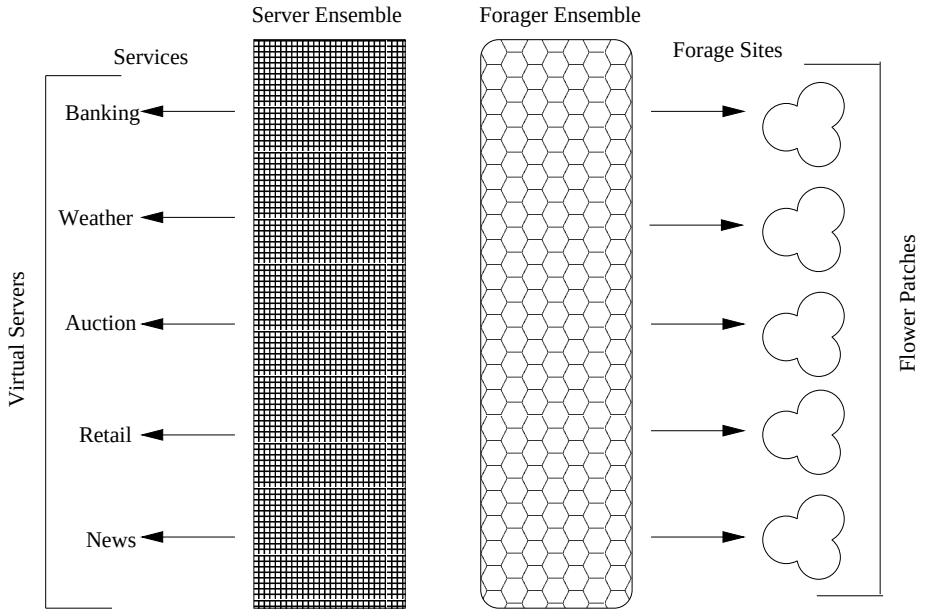

Figure 4. Server and forager orchestration problems.

5. Homology: An Internet Hosting Centre and Honeybee Colony

A cursory glance at the server and forager orchestration problems, as depicted in Figure 4, would suggest a similarity between the two problems, at least on a superficial level. The server ensemble in an internet hosting centre is analogous to forager bees in a honeybee colony whilst the service request queues pertaining to co-hosted internet services are analogous to sources of nectar to be exploited profitably. Moreover, an examination at finer granularity revealed a remarkably close mapping between the two problems, as illustrated in Table 1. The strong homology between the server and the forager orchestration problems motivated our biomimetic approach.

An internet hosting centre co-hosting multiple internet services on its finite server ensemble is analogous to a honeybee colony with a certain number of bees foraging at multiple forage sites in the surrounding countryside. The amount of time required by a server to earn revenue from a hosted internet service will depend on the service characteristics as well as total number of servers servicing a given request queue. If more servers are orchestrated than necessary to a service, the revenue rate will degrade due to each server having to wait longer to find a request to serve. Similarly, the amount of time needed by a forager to return with a nectar payload depends on characteristics of the forage site as well as the number of foragers collecting nectar there. The rate of nectar payload being returned to the colony will degrade if overly many foragers are orchestrated to the site since each forager will take longer to find flowers whose nectar has been replenished but not yet harvested.

Streams of requests arriving over the Internet at a hosting centre for co-hosted internet service are buffered by respective service queues. An equivalent in a honeybee colony are available flower patches (forage site) in the surrounding countryside. This is

Table 1. Parallels between Server and Forager orchestration problems.

Internet Hosting Centre	**Honeybee Colony**
An ensemble of N servers.	An ensemble of N nectar foragers.
Unit of orchestration: Single server.	Unit of orchestration: Single nectar forager.
Co-host M internet services to be accessed by requests.	M distinct flower patches in the surrounding countryside.
A group of servers (virtual server) with the same internet service application serving requests from a specific service queue.	A group of nectar foragers collecting nectar at a specific flower patch.
Time to service a request will be dependent on the internet service application.	Time to travel will be dependent on the location of the flower patch.
(*)Time taken by a server to find a request to serve at an internet service application increases if number of servers in the group increases	(+)Time taken by a forager to collect nectar at a flower patch increases if number of foragers in the group increases
(*)A server orchestrated to another internet service incurs downtime due to purging of current and installation of new internet service application/data	(+)A forager orchestrated to another flower patch incurs downtime due to time spent learning the location and successful discovery of the patch
(*)An internet service application and its requests have a *value-per-request*	(+)A flower patch has quality of nectar (sugar concentration)
Variable rates of requests arrival and balking behaviour	Variable rates of flower patch quality, density and replenishment
A server repeatedly serving requests of the same type results in improved response time	A forager repeatedly foraging on flowers of the same type results in improvement in its ability to extract nectar
The flagged(*) items change unpredictably over time for each service	The flagged(+) items change unpredictably over time for each flower patch

intuitive since servers collect SLA revenue for serving requests from the service queues whilst foragers collect nectar from flowers at the forage site. The aptness of the analogy is further demonstrated by similarities in variability of request stream behaviour and flower patches volatility. Request streams are variable due to fluctuating request arrival rates and balking behaviour of users waiting for service whilst flower patches are variable due to fluctuations in micro-climate, quality, density and nectar replenishment rates.

In an internet hosting centre, a virtual server hosting an internet service is composed of one or more servers collecting revenue from its service queue. The unit of augmentation or reduction is a single server. In a honeybee colony this is equivalent to the set of forager bees collecting nectar from a specific flower patch. The unit of augmentation or reduction is a single forager bee. When a server is orchestrated to leave a given virtual server and join another virtual server, it incurs a downtime penalty since it undergoes a repurposing process whereby current internet service application and associated data are purged and new internet service application and associated data are installed. During the repurposing process, the said server makes no contribution to the revenue earning endeavour of the hosting centre. Similarly, when a forager is enticed to forage at a particular flower patch, she follows a waggle dance that signals its location. However, it typically takes her several attempts to successfully find the new patch [12]. During the time spent locating the flower patch, she does not contribute to the nectar collecting endeavour of the colony [10]. There is an additional lovely parallel between the two. If a honeybee makes repeated visits to a flower patch, she improves her ability to extract nectar from that particular kind of flower [8]. If a server repeatedly serves customers of the same type, its service speed for that type of customer tends to increase, because of the storage of data in its caches [13].

Each virtual server hosting an internet service in a hosting centre has an associated SLA contract with the service owner which defines value-per-request-served among other components whilst each flower patch has an associated nectar quality (mol./L sugar concentration). Thus, a server in a hosting centre collects some value-per-request-served whilst a forager bee in a honeybee colony collects some mol./L sugar concentration from the nectar.

Finally, the time to service a request in a hosting centre depends on the application of the internet service and the time to find a request to serve will be a function of the number of servers serving a given request queue as well as the request arrival pattern. In a honeybee colony, the travel time to a specific forage site depends on its location in the countryside and the time needed to collect nectar will be a function of the number of foragers foraging at a given forage site as well as the rate of nectar replenishment of the flowers at that site. In a hosting centre, different internet services will have different values of service-time-per-request and value-per-request-served as per negotiated SLA contract. In a honeybee colony, different forage sites in the countryside will have different values of forager-round-trip-time and nectar quality.

Let us explore the accuracy of this aspect of the homology. In the case of a honey bee colony, let $f_i(x_i)$ denote the number of nectar loads returned per minute by a total of

x_i bees devoted to foraging at the i^{th} flower patch. For simplicity we suppress the index i and normalise the nectar quality. Let T denote the travel time from the patch to the hive and back (including nectar unload time). Hence, of the $\frac{x}{f(x)}$ minutes required by a single bee to collect and deliver a nectar load, the fraction $1 - \frac{Tf(x)}{x}$ is spent collecting at the patch. Therefore (on average) $x - Tf(x)$ bees are at the flower patch at any time, and the forage time of a single bee is $\frac{1}{f(x)} = T + w(x - Tf(x))$ minutes, where $w(y)$ denotes the (steady state) time to collect a nectar load if y bees are actively foraging there. One may recover $f()$ from $w()$ via

$$f\left(x + \frac{xT}{w(x)}\right) = \frac{1}{T + w(x)}. \tag{1}$$

In the case of an internet hosting centre, $f(x)$ denotes the revenue rate per minute of x servers in the virtual server V serving requests from service queue Q, and T denotes the service processing time for V. In this case equation 1 represents the revenue-earnings time of a single server, where $w(y)$ denotes the (average) waiting time for one server amongst y to acquire a customer from queue Q.

Thus the cycle times of both bee and server have the same functional description. We now show that the functions $w()$ and hence $f()$ have similar qualitative behaviour in the two cases. For the honey bee colony, $w(1)$ is the nectar load collection time from the patch with one bee collecting nectar. Now, if two bees are collecting nectar from the flower patch, then there is a slight possibility of interference due to the activity of each bee. One bee will collect nectar from a specific flower whose nectar will be replenished with some delay. Soon afterwards, the other bee may attempt to collect nectar from the said flower with an outcome that it will fail to find any nectar and will lose time to try another flower. Therefore, $w(2) \gtrapprox w(1)$ because of the slight interference effect. If a third bee were collecting nectar as well, the other bees would experience an additional interference effect because she would deplete additional flowers (We are neglecting the 2^{nd} order effect of interference reducing interference rates.) Thus $w(3) \gtrapprox w(2)$. By the same reasoning, for small values of y (relative to total flower patch nectar replenishment rate $\mathcal{N}$ loads/minute) we have $w'(y) \gtrapprox 0$.

In the case of two servers at a virtual server of the internet host centre, each takes an available request from the service queue, but a slight interference effect can be triggered when Q contains exactly one request. Thus $w(2) \gtrapprox w(1)$. More generally, for small values of y the interference effect increases in y but is not large since it can only occur if at least one but fewer than y requests are present in the service queue. Thus, as with the honey bees, we have $w'(y) \gtrapprox 0$. At the other extreme, if $\frac{y}{w(1)} \gg \mathcal{N}$, the foraging capacity will essentially saturate the patch's nectar production. Thus $w(y) \approx \frac{y}{\mathcal{N}}$. In the host centre, if $\frac{y}{w(1)} \gg \mathcal{N}$ the virtual server will lose essentially zero requests to balking, and thus again $w(y) \approx \frac{y}{\mathcal{N}}$. At intermediate values $y \gtrsim w(1)\mathcal{N}$, it is not so clear how the function behaves. Interference effects force $w(y) > w(1)$, but nectar harvesting (respectively customer service) should be close enough to saturation that one additional forager (respectively server) increases nectar(respectively revenue) collection

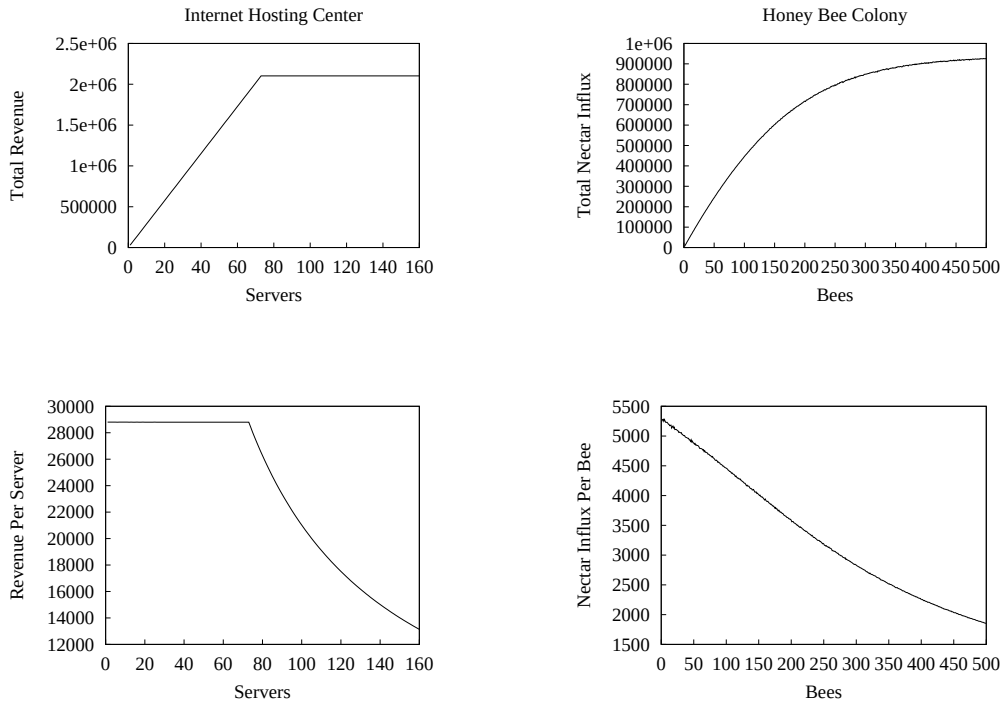

Figure 5. Revenue and Nectar influx functions of a honeybee colony foraging at a single flower patch and an internet hosting centre serving requests of the same type at a hosted service.

only slightly, thus $[\frac{y}{w(y)}]' \gtrsim 0$ whence $w'(y) \lesssim \frac{w(y)}{y}$.

To test the above analysis, we developed a simulation model of bees foraging at a flower patch, since we could not obtain empirical data. Flowers were modelled as points in a grid, which replenish nectar at some rate; bees were modelled as travelling randomly from grid cell to neighbouring cell, until a full load is collected. Other model parameters included number of flowers in the patch, nectar extraction time, and round trip time. We performed a simulation run for each increment in allocation of a bee to the flower patch over a 24 hour time horizon and collected performance data. The plot of total nectar influx and nectar influx per bee for varying allocations of bees to the flower patch is depicted in Figure 5(Honeybee Colony).

For the case of the internet host, we ran the hosting centre simulation model [14] over a 24 hour time horizon with one virtual server and an inhomogeneous Poisson request stream distribution, for each increment in server allocation to the virtual server. Figure 5(Internet Hosting Centre) illustrate the total revenue and revenue per server for varying allocation of servers. The simulations confirm our analysis. The qualitative behaviours of the functions are similar in the two cases. However, in the internet centre data the functions appear nearly piecewise-linear with a single break point, whereas the slopes in the honey bee data do not change rapidly. In other words, the second derivative spikes much more in the internet centre case. The homology fails at this level, although

admittedly past the scope of our empirical observation.

Thus there are many close parallels between the problems. Based on our simulation tests, the homology begins to lose accuracy at the level of the nature of the interference effects. The net revenue and net nectar influx functions do have the same rough qualitative behaviour, increasing fairly rapidly and then flattening out. However, the nectar influx flattens out considerably more gradually. The functions and their first derivatives are not greatly different, but the second derivatives are.

6. Biomimetic Server Ensemble Orchestration Algorithm

A self-organising server ensemble orchestration algorithm for internet hosting centres follows readily from the homology that we have exposed. As in the case of the forager orchestration in a honeybee colony, the biomimetic server ensemble orchestration algorithm must acquire information on an ongoing basis in respect of prospective co-hosted internet services and request load on their service queues as illustrated in Figure 6. In order to gather relevant information, we mimic signals, cues as well as other mechanisms of a honeybee colony for an internet hosting centre. Each server in a hosting centre is either a forager or a scout server. The dance floor is represented by an advertisement board. A waggle dance of some duration is mimicked by an advertisement whose posting appears on the advertisement board for some time length. Furthermore, a flower patch location is represented by a virtual server (service) identifier whilst waggle dancing and following a waggle-dance are mimicked by advertisement posting and advertisement reading respectively. The biomimetic server orchestration in an internet hosting centre relies on each server generating information about its own service queue and making it globally available whilst all servers in the ensemble have the opportunity to view this information and act upon it. Basically, a server services requests and posts advertisements of length varying with revenue potential. It regularly checks its own revenue rate against the hosting centre's and adjusts its probabilities of advertising or re-orchestration. To re-orchestrate, a forager server will randomly select an advertisement and migrate to the corresponding virtual server, whilst a scout server will migrate to a random virtual server.

7. Application of *Swarm Intelligence:* A Small Survey

Swarm intelligence methodology has been applied in a variety of domains, with varying degrees of success. Here we briefly survey some of these applications, looking for common patterns that may lead to success or failure.

Schoonderwoerd *et al.* [15] developed an ant-colony inspired pheromone-laying algorithm to select the next path for a message in a telecommunication network. The underlying issue in this system is the selection of a specific path between each source-destination pair so as to balance the load on the different links in the network. Other researchers have explored ant colony inspired methods for "connectionless"

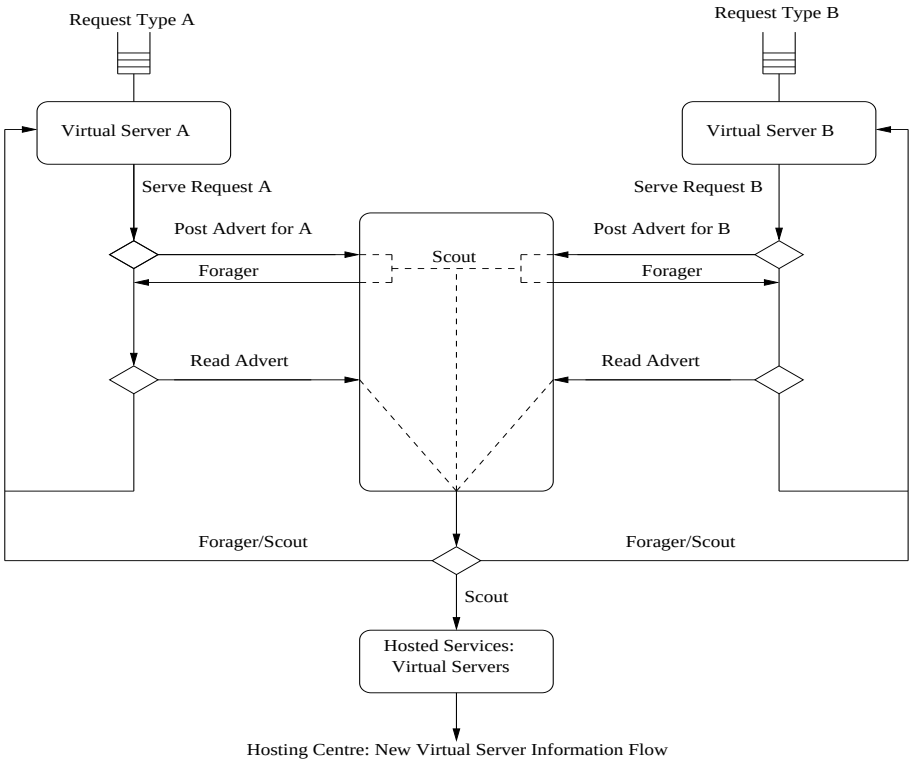

Figure 6. The information flow and decision processes in an internet hosting centre.

network routing problems, such internet routing, in which there is no specific path (see Merloti [16] for a fuller explanation). Network links can experience unpredictable congestion or failure. The routing problem is even more dynamic in these connectionless domains. Di Caro and Dorigo [17] imitate ant pheromone trail laying and following behaviour to dynamically route packets on the internet. The resulting algorithms are simple and intuitive. Di Caro and Dorigo report that their performance is superior to more complex alternatives in terms of throughput and packet delay. Overall, network routing appears to be one of the leading successes in swarm intelligence applications. Other promising applications include image clustering and segmentation [18], fault-tolerant vehicle scheduling [19], and factory control/scheduling [20, 21, 22].

On the other hand, swarm intelligence has encountered some notable failures. For example, Bonabeau and Meyer [23] report that Enron established a swarm intelligence management model in its energy trading business, which produced a financial disaster of the first magnitude. The ACO (ant colony optimisation) travelling salesman algorithms reported in [24, 25, 26] are heavily reliant on a different optimisation method, neighbourhood search, for their success, but even so they are not remotely competitive with operations research/computer science methods [27, 28]. Shortest path algorithms in computer science [29], likewise, outperform any reinforcement or brute force parallel swarm intelligence method by an order of magnitude.

What can we learn from these cases? One clue comes from the successes and failures discussed in the non-technical book "The Wisdom of Crowds" [30]. From a mathematical

view, the point of the book is that the law of large numbers is true. That is, the sample mean of a large number of independent observations converges to the true mean. If a large number of people are asked independently to estimate the weight of an ox or the number of cherries in a bowl, the sample mean is apt to be quite accurate. On the other hand, mobs are infamous for their collective stupidity. In the Enron case, energy traders recruited hundreds of others to engage in the same activity, but the individual traders did not receive independent objective feedback as to the profitability of their actions. In comparison, each individual packet in network "knows" how successful it is in arriving quickly to its correct destination. Despite the stochasticity of the transit times, the aggregate measure, namely the pheromone trail, which mathematically speaking is simply a (time-weighted) sample mean, is apt to be quite accurate. The same kind of accuracy is obtained through the aggregation of sampling information in our honeybee algorithm, as it is in natural honeybee colonies. See Passino [31] for a similar insight into aggregation of noisy or stochastic information.

Related to this feature is the presence or absence of varying local information. This can be thought of as varying information in space rather than in time (dynamicism). We suggest that variability in (possibly virtual) space is another feature which tends to make swarm intelligence appropriate as a solution approach. This is because each of the many individual agents can sense and react to its local information, creating a non-homogeneous, locally adaptive policy. For example, the wireless network domain where mobile *ad hoc* networks are spatially and temporally formed by autonomous collection of mobile devices [32]. If both of these features are present, a swarm intelligence solution can be rapidly adaptive to changing local conditions. Such situations seem particularly appropriate to this approach.

Another telling feature in these cases is whether or not the parallelism is real or simulated. In the ACO algorithms for combinatorial optimisation problems [26], the insect colony members are simulated on a computer. The algorithms do not actually run on a myriad of small cheap independent processors. In contrast, natural insect swarms do contain thousands or millions of individual insects, communications networks do handle a myriad of individual packets on a large set of links and nodes, and our internet hosting centre does contain many autonomous servers. We suggest that swarm intelligence is more likely to be successful if the parallelism is inherent in the system to be optimised, rather than artificially added on.

Finally, we observe that swarm intelligence seems to be more successful when it implements at least one balancing (negative) feedback loop. If swarm intelligence offers only a positive feedback loop, i.e. of the form "if this activity is good, do more" then it provides only a weak and imprecise thrust towards optimality. For example, positive feedback is an inefficient way to maximise a linear function [33]. In contrast, congestion creates negative feedback loops in networks, so that in an optimal configuration different packets traverse different routes. Similarly, in an optimal server allocation different servers host different web services.

8. Conclusions

In this paper, we probed the homology between honey bee colony forager allocation and internet host server allocation that inspired a new Biomimetic server orchestration algorithm for management of internet hosting centres. We exposed many close parallels between the problems. We also found a qualitative difference which appears at the rather fine level of the 2^{nd} derivatives of the nectar and revenue influx functions. The Biomimetic algorithm performs very well in a variety of circumstances [34]. We suggest that the effectiveness of the algorithm is derived from the tightly coupled homology between the problems in the natural and human domains.

Based on our experience and on a small survey of other cases, we conjecture that the following conditions promote the suitability and success of swarm intelligence applications: inherent parallelism in the problem domain, presence of at least one balancing (negative) feedback loop, noisy or variable data — often temporally or spatially local — that tends to be accurate in the aggregate, and the need for rapid and effective responsiveness to greatly changing circumstances.

Acknowledgments

We would like to thank Professor Tom Seeley at Cornell University for providing invaluable insights into the workings of honeybee colonies.

References

[1] Nakrani S and Tovey C 2003 *Proceedings of 2nd International Workshop on The Mathematics and Algorithms of Social Insects (Atlanta)* p 115-122
[2] Brewer E 2000 `http://db.cs.berkeley.edu/~jmh/cs262`
[3] Arlitt M and Jin T 1999 `http://citeseer.nj.nec.com/article/arlitt99workload.html`
[4] Chase J Anderson C Thakar P and Vahdat A 2001 *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*
[5] Cherkasova L 1999 *Technical Report HPL-1999-52(R.1) (HP Laboratories)*
[6] Jayram T Kimbrel T Krauthgamer R Schieber B and Sviridenko M 2001 *Proceedings of the STOC(Hersonissos, Crete, Greece)*
[7] Markoff J 2003 *New York Times (13th November edition)*
[8] Seeley T 1995 *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies* (Harvard University Press)
[9] Bonabeau E. Dorigo M. and Theraulaz G. 1999 *Swarm Intelligence: From Natural to Aritificial Systems* Oxford University Press
[10] Seeley T Camazine S and Sneyd J 1991 *Behaviour Ecology and Sociology* p 277-290
[11] Bartholdi(III) J Seeley T Tovey C and Vate J 1993 *Theoretical Biology*
[12] Seeley T 2003 *Personal communication*
[13] Pai V Aron M Banga G Svendsen M Druschel P Zwaenepoel W and Nahum E 1998 *Proceedings of 8th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII) (San Jose)*
[14] Nakrani S and Tovey C 2004 *Adaptive Behaviors* **12** 3–4 p 223-240
[15] Schoonderwoerd R Holland O Bruten J and Rothkrantz L 1996 *Technical Report (HP Laboratories (HPL-96-76))*

[16] Merloti P 2004 `http://www.merlotti.com/EngHome/Computing/AntsSim/ants.htm`
[17] Di Caro G and Dorigo M 1998 *Artificial Intelligence Research* **9** p 317-365
[18] Ramos V Muge F and Pina P 2002 *Proceedings of the 2nd International Conference on Hybrid Intelligent Systems (Santiago)*
[19] Bullnheimer B Hartl R and Strauss C *Technical Report (University of Vienna (10/97)*
[20] Cicirello V and Smith S 2001 *Proceedings of the 5th International Symposium on Autonomous Decentralized Systems (ISAD-2001)*
[21] Cicirello V and Smith S 2001 *Proceedings of the Workshop on AI and Manufacturing (IJCAI-01)*
[22] Cicirello V and Smith S 2001 *Proceedings of the 5th International Conference on Autonomous Agents (Agents 2001)*
[23] Bonabeau E and Meyer C 2001 *Harvard Business Review (May edition)*
[24] Dorigo M and Gambardella L 1997 *BioSystem* 43 p 73-81
[25] Colorni A Dorigo M Maniezzo V 1991 *Proceedings of the European Conference on Artificial Life (ECAL91, Paris)*
[26] Colorni A Dorigo M Maffioli F Maniezzo V Righini G and Trubian M 1996 *International Transaction in Operational Research* **2** 1 p 1–21
[27] Aarts E and Lenstra J (Eds) 1997 *Local Search in Combinatorial Optimization* (John Wiley and Sons Ltd) p 215–310
[28] Applegate Bixby R Chv'atal V and Cook W 1998 *Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung* p 645–656
[29] Cormen T Leiserson C and Rivest R 1990 *Introduction to Algorithms* (The MIT press) p 329–355
[30] Surowiecki J 2004 *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations* (Random House Inc.)
[31] Passino K 2005 *Biomimicry for Optimization, Control, and Automation* (Springer-Verlag)
[32] Sesay S Yang Z and He J 2004 *Information Technology* **3** 2 p 168–175
[33] Goldberg D 1989 *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley Pub. Co.)
[34] Nakrani S and Tovey C 2006 Resilience of Honeybee inspired Server Orchestration in Internet Hosting Centre *Preprint*