The Effect of Load on Agent-Based Algorithms for Distributed Task Allocation

Harry Goldingay, Jort van Mourik

Non-linearity and Complexity Research Group, Aston University, Aston Triangle, Birmingham, B4 7ET, UK

Abstract

Multi-agent algorithms inspired by the division of labour in social insects and by markets, are applied to a constrained problem of distributed task allocation. The efficiency (average number of tasks performed), the flexibility (ability to react to changes in the environment), and the sensitivity to load (ability to cope with differing demands) are investigated in both static and dynamic environments. A hybrid algorithm combining both approaches, is shown to exhibit improved efficiency and robustness. We employ nature inspired particle swarm optimisation to obtain optimised parameters for all algorithms in a range of representative environments. Although results are obtained for large population sizes to avoid finite size effects, the influence of population size on the performance is also analysed. From a theoretical point of view, we analyse the causes of efficiency loss, derive theoretical upper bounds for the efficiency, and compare these with the experimental results.

Keywords: Adaptive systems, algorithms, distributed decision-making, response thresholds.

1. Introduction

Distributed systems form an important field of active research with an ever increasing number of applications such as distributed heterogeneous computing [15], and mobile sensor networks [22, 28]. For an efficient coordination of of the sub-systems, distributed resource and task assignment is a core problem within distributed systems, in fact Lerman et al. [21] describe it as "a key functionality". Although many current solutions to this problem focus on a centralised approach, there is growing feeling that the size of some distributed systems necessitates a decentralised approach as in large scale distributed systems, communication costs can significantly limit performance [16]. Furthermore, Chevaleyre et al. [5] state that "future manufacturing systems ... must possess such attributes as decentralisation, distribution, autonomy, adaptability, and incomplete information handling".

The common algorithms for distributed task-allocation can be broken down into three categories: simulation-, AI-, and agent-based [33]. When presented with an event, however, simulation and AI-based approaches centrally control their sub-systems after evaluating potential responses through a computational model of the system. While this can lead to good coordination, on large scale, complex problems, scalability, along with the difficulty of designing and evaluating the model, makes these approaches impractical [33]. Agent-based approaches, however, are particularly attractive when designing scalable systems due to being inherently decentralised. Agents make decisions based on local interactions (either directly with each other, or through their environment) and good algorithms allow them to adapt, enabling them to coordinate through self-organisation [29]. This means that global behaviour emerges from simple, local interactions, providing good scalability while avoiding the need for complex computational models.

The two most successful types of agent-based task allocation algorithms are threshold- and market-based algorithms. Threshold-based algorithms were developed from the threshold model of task allocation in insect colonies [3, 32]. In this model tasks are categorised into a set of *types*. An individual chooses to engage in a task by probabilistically comparing some environmental stimulus for the task with some internal preference for the task's type. Individuals adjust their preferences through self-reinforcement in order to react to levels of demand. While this was originally a model of biological behaviour, the link between social insects and multi-agent systems is well established

Email addresses: goldinhj@aston.ac.uk (Harry Goldingay), vanmourj@aston.ac.ul (Jort van Mourik)

[1], and the threshold model has since inspired decentralised task allocation algorithms in the engineering domain (i.e. [4, 16, 19, 24, 25, 26]).

Markets are another area which have provided inspiration for the design of self-organising multi-agent systems and have been used in a wide range of applications [9]. Market based algorithms treat their elements as self-interested individuals with goals. Groups of individuals use market-like mechanisms, typically in the form of an auction, to decide who fulfils these goals, with bids based on both the individual's desire and ability to complete their goal. Individual goals are designed to further the overall goals of the system, meaning that these auctions can allow agents to coordinate and produce desirable system-wide behaviour. While this approach does require some level of communication between agents (hence it is not applicable to all problems), if desired, this can be kept at the local level, avoiding the scalability issues associated with centralised control.

Current comparative studies of threshold- and market-based algorithms tend to focus on single test problems, with fixed parameters [6, 14, 24, 33]. They compare quantitative performance to establish which algorithm performs best in their setting. This approach would be sufficient if we had a test problem which could accurately predict algorithmic performance on real world problems. However, Xiang and Lee [33] claim that current task allocation test problems do not give accurate results as they are oversimplified, and try to address this problem by designing a more realistic problem model. However, this new problem still contains unrealistic assumptions and is specific to the job shop scheduling problem [33]. Further, it is not clear that performance on one specific real world problem generalises well to problems with different conditions and constraints. Hence, it is important to consider the reaction of an algorithm to qualitative trends. In particular, it is important to consider under what types of conditions an algorithm loses performance, and what features of the algorithm cause this. Kalra and Martinoli [17] take this approach, investigating the effect of communication range and state estimation on threshold- and market-based algorithms. They find that market-based algorithms perform better under perfect state estimation and good communication conditions, but break down in poor conditions.

When comparing algorithmic behaviour, a key concept to consider is *load*. Here, load does not refer to a specific quantity, but rather to the general pressure on an agent to complete tasks in order for the system to function efficiently. A simple way of representing load is with the ratio of agents to tasks as, clearly, the more agents there are to each task, the less pressure there is on each individual agent to complete a task. Since other factors influence the difficulty of task completion, however, we also consider other representative quantities (of load). Clearly, load is an important concept as, especially in an engineering context, it is costly and inefficient to employ more resources (agents) to complete a task to a given standard. Hence, robustness to high load levels is of paramount importance when selecting solution methods for a given problem. Partial results on the effect of load are given by Cicirello and Smith [7]. They investigate performance under both high and low "set up times", effectively time penalties paid by an agent upon violating a constraint. As these penalties can render agents inactive, they can be seen as a type of load on the system. However, there is no systematic, qualitative, investigation into the effect of load on agent-based algorithms for distributed task allocation.

As mentioned previously, there is no existing problem which perfectly predicts the performance of task allocation algorithms in a real world setting. Therefore, we need a test problem which captures the key features of a distributed task allocation problem. Ideally, we require it to be:

- complex enough that agents must make trade-offs to achieve good performance (i.e. no trivial best algorithm).
- simple enough to analyse theoretically.
- flexible enough to test the algorithms in a wide range of settings.
- computationally viable when testing large scale systems.

As our focus is on comparing the task selection behaviour of threshold- and market-based algorithms, we neglect other details, which would reduce the generality of our problem. In order to do so, we apply existing threshold- and market-based algorithms to a problem of distributed task allocation based on a mail retrieval problem in which agents travel to distinct locations (cities) at which they must choose between various tasks (types of mail to process). It has been used in various comparative studies of algorithms for distributed task allocation, presented either as a mail processing [25, 26, 12, 13], or truck painting problem [4, 19]. It is a flexible problem which allows for a comprehensive investigation of the behaviour of candidate algorithms. As the version of the problem we study is both large scale and

only admits local information [13], it is ideal for studying the behaviour of decentralised, self-organising algorithms. We stress that the problem is intended as a prototype for general distributed task allocation rather than as a realistic model for either mail processing or truck painting.

In this paper, we conduct an analysis into the effect of load on the performance of threshold- and market-based algorithms. To this end, we systematically adjust several parameters in order to vary the effective load on the system. The results are mainly given in terms of the efficiency (average mail processed per agent per time step). We theoretically analyse the different causes of *loss* of efficiency, and derive the theoretical maximum for the efficiency to which the the numerical results are compared. As the algorithms depend on parametrised functions, we optimise these parameters using a nature inspired particle swarm optimisation (PSO) algorithm, which allows us to compare algorithmic behaviour unbiased by parameter choice. We show that, under a variety of measures of load, market-based algorithms outperform threshold based algorithms under low load levels, while threshold-based algorithms are able to maintain performance under high load levels where purely market based algorithms break down catastrophically due to their inability to reject tasks. Finally, by selectively combining the two approaches, we present a novel hybrid algorithm that is able to match or exceed the performance of the market-based algorithm while maintaining the robustness against high load of the threshold-based algorithm.

The paper is organised as follows: in next section 2, we introduce the problem and, in 3, the various strategies to solve it. In section 4, we perform a theoretical analysis that in the large system limit under certain conditions allows us to provide a theoretical upper bound for the performance of any algorithm. In section 5, we present and discuss the numerical results and compare them with the theoretical bounds. Finally, in section 6, we summarise our main findings, discuss the limitations of the current setting, and give an outlook to future work.

2. Problem Definition

2.1. The Mail Processing Problem

The problem comprises a set of N_c mail producing cities, each of which is capable of producing and storing one batch each of N_m different mail types. Each city c has a fixed-length vector of mail $\boldsymbol{w}_c = (w_{c,1}, .., w_{c,N_m})$ where $w_{c,m}$ is the waiting time of the batch of mail type m[1]. A piece of mail with high waiting time is representative of a high priority task. Note that $w_{c,m} = 0$ indicates that there is no batch of that type present, either because no such batch was produced, or because another agent has already taken it. If a city is not storing a batch of mail type m at the beginning of time-step t, a new one is produced with probability $\pi_m(t)$. Upon production of a batch of mail type m, its waiting time is initialised to $w_m = 1$, and at the end of each time step the waiting times of remaining batches of mail are increased by 1.

To process the mail, we have a set of N_a mail processing centres. Each centre has one associated mail collection agent whose task it is to travel to a city and return with a batch of mail for processing. Each type requires a different processing method and at any point in time the processing centre of agent a is specialised in one specific type σ'_a. The centre can process a batch of this type efficiently in a time t_p. However, in order to process a batch of a different type m, the centre must undergo a **changeover** $\sigma'_a \to m$ which takes a time $t_c > t_p$ (including the processing of the batch).

In order to reduce the direct impact of these changeovers, each centre has a **mail queue** in which it can temporarily store mail while processing other batches. This queue is capable of storing up to L_q batches of mail and, while there is space, the agent will continue to collect mail. A centre must process the mail in its queue in the arrival order, such that all the freedom in the system is concentrated in the behaviour of the collection agents. Therefore, we define the **effective specialisation** σ_a of the agent as the last collected mail type, because σ'_a will be σ_a by the time the next collected mail is processed.

All mail types represent distinct tasks in which agents engage upon uptake of a batch. Agents visit cities and, once there, act as determined by their algorithms. Specifically, for a threshold-based algorithm agents are individually allowed to examine mail at a city, accepting or rejecting each batch in turn based on their response thresholds. For market based algorithms *all* agents at a city are offered each piece of mail in turn and must submit a bid determined by their bidding function, with the highest bidder taking the batch of mail. Once an agent selects a piece of mail, it is then deposited in the queue of the agent's processing centre, $\boldsymbol{q}_a = (q_{a,1}, .., q_{a,L_q})$, which can store a backlog of up to

[1] Note that this means a city can contain between 0 and N_m batches of mail, but can never contain multiple batches of the same mail-type.

L_q batches of mail. Agents with a full queue (i.e. $q_{a,L_q} \neq 0$) cannot take any more mail, and are **inactive**, while agents with space in their queues are **active**. Formally, time evolves in discrete steps of $\Delta t = 1$, and at each step, the problem proceeds as in algorithm 1.

Algorithm 1 An iteration of the mail processing problem at time t.

for agent $a = 1, ...N_a$ **do**
  if queue isn't full (if $l_a \neq L_q$, where q_{l_a} is the last non-zero entry in $\boldsymbol{q}_a$) **then**
    Choose a city (add a to the set Ψ_c) for random $c \in \{1, ...N_c\}$
  end if
end for
for city $c = 1, ...N_c$ **do**
  for mail type $m = 1, ...N_m$ **do**
    if mail of type m is not already at city c (i.e. if $w_{c,m} = 0$) **then**
      produce mail of type m (set $w_{c,m} = 1$) with probability $\pi_m(t)$.
    end if
  end for
  Agents in Ψ_c act at city c, **details are algorithm dependent**
  for mail type $m = 1, ...N_m$ **do**
    if mail of type m is already at city c (i.e. if $w_{c,m} \neq 0$) **then**
      increase waiting time of mail type m (set $w_{c,m} = w_{c,m} + 1$).
    end if
  end for
end for
for agent $a = 1, ...N_a$ **do**
  if agent a accepted a piece of mail (type denoted m) **then**
    if mail matches effective specialisation (if $m = \sigma_a$) **then**
      add mail to the processing queue ($q_{a,l_a+1} = t_p$)
    else
      add mail to the processing queue with a penalty ($q_{a,l_a+1} = t_c$)
      switch effective specialisation to account for the changeover ($\sigma_a = m$)
    end if
  end if
  if agent a has mail in its queue (if $q_{a,1} \neq 0$) **then**
    Continue to process the first item in the queue
    if the agent has finished processing a batch of mail (if $q_{a,1} = 0$) **then**
      move remaining batches of mail up in the queue
      (set $q_{a,l} = q_{a,l+1}, l = 1, ..., L_q - 1$ and $q_{a,L_1} = 0$)
    end if
  end if
end for

A threshold-based algorithm has previously been employed to solve a version of this problem in which agents were capable of remembering, and returning to, previously discovered cities [12]. This memory, also stigmergic in nature, adds an layer of self-organisation to the system and allows for performance very close to that of ideal centralised control without the need for any communication. However, it is easy to imagine a real-world problem in which memory of locations is impractical. Task locations may be non static, rendering memories of past locations obsolete, or the environment may be difficult to navigate, making the problem of agent navigation more difficult than the initial problem of task allocation. Therefore, it is important to consider a version of this problem in which agents are incapable of preferential choice of task site.

As we wish to capture only the essential features of the problem, we do not impose a specific topology on the cities and assume that any agent can travel to any city, and return, within a single time-step. While this assumption is somewhat unrealistic given the large scale of the system, there is no single topology which is representative of

all real world problems. Further, the introduction of topology would require agents to use some routing algorithm. Again, there is no single routing algorithm ideal for all distributed task allocation problems. As these additions would increase the complexity of the problem without increasing the generality of our results, we exclude them.

We wish to study our algorithms with varying requirements on flexibility, and we consider two different types of environment:

- A static environment, in which a city automatically replaces each taken batch of mail at the end of each time step.
- A dynamic environment, in which the probabilities of production of batches of the mail types varies over time.

In the dynamic environment we have chosen to vary the probability of taken mail batches being replaced in a sinusoidal fashion. However, the exact form of the function is not critical, we merely want an environment in which continuous adaption is required. The probability of creating a taken batch of type m at the end of cycle t is given by:

$$\pi_m(t) = \begin{cases} 1, & \text{static} \\ \frac{1}{2}[1 + \sin(\frac{t2\pi}{\xi} - \frac{m2\pi}{N_m})], & \text{dynamic} \end{cases} \tag{1}$$

where ξ is the wavelength.

Our aim is to maximise the efficiency (mail processed per agent per time step) of the process. As this is equivalent to minimising the loss of efficiency, it is important to identify the different mechanisms that lead to efficiency loss. If an agent of effective specialisation σ_a fails to process mail during an iteration, this can be categorised into four cases:

($\boldsymbol{\ell}$.1) The agent is inactive due to a full mail queue at its processing centre.
($\boldsymbol{\ell}$.2) Mail type σ_a is available at the city, but the agent rejects all mail.
($\boldsymbol{\ell}$.3) Mail type σ_a is unavailable at the city and the agent rejects all mail.
($\boldsymbol{\ell}$.4) There is no mail at all available by the time of the agent's action.

We will use $\boldsymbol{\ell}$.1-4 to denote the average losses in their respective categories per agent per time step. As an agent can either take mail (adding to efficiency) or fail to take mail in exactly one of the above ways, we have

$$\text{efficiency} = 1 - - \sum_{i=1}^{4} \boldsymbol{\ell}.i \tag{2}$$

While it is possible to minimise $\boldsymbol{\ell}$.4 by increasing $\boldsymbol{\ell}$.1-$\boldsymbol{\ell}$.3 (i.e. by lowering the overall acceptance rate of mail), this is clearly not ideal. In particular, $\boldsymbol{\ell}$.3 and $\boldsymbol{\ell}$.4 are due to the non-uniform number of agents visiting cities. In the current model, we have no control over this and focus mainly on the other sources. Clearly $\boldsymbol{\ell}$.2 is unnecessary, as the uptake of mail of its own specialisation has no negative consequences for an agent. One should note that $\boldsymbol{\ell}$.1 and $\boldsymbol{\ell}$.3 are finely balanced against each other as a greater uptake of mail of non-specialised types leads to an increase in agents with full queues.

As the only freedom lies in the behaviour of agents at cities, one can only minimise these loss sources by selecting an agent rule-set which has the optimal balance between loss sources under given circumstances. In particular, agents should take mail of the same type on consecutive occasions with the high probability, minimising changeovers and the probability of their queues filling up. However, agents should retain some flexibility and not compromise their ability to adjust to the current demand level (giving a long term advantage) to avoid a single changeover (a short term penalty).

2.2. Load

As previously mentioned, load is the effective pressure on agents to complete tasks in order for the system to have good performance. As agents are homogeneous, and tasks are assigned to single agents, this pressure is directly related to the ratio of active agents to tasks available. In the current setting, this ratio can be adjusted directly, by changing the number of agents for a fixed number of tasks, or indirectly, by increasing the probability that agents become inactive. We consider three measures of load:

- The ratio of agents to number of possible mail batches $R_{a/m} = \frac{N_a}{N_m \times N_c}$ is the direct measure of load.
- The changeover time t_c changes the frequency with which an agent can undergo changeovers while remaining active.
- The number of mail types N_m changes the probability of a changeover.

3. Algorithm Details

The performance of our task allocation algorithms is determined by the agents' behaviour at cities. To highlight the possible extremes of algorithms, we imagine two rules for agent behaviour. In the first, agents "greedily" take mail if available thus ensuring cities to be maximally served in the short term. In the second, agents are completely "selective", refusing to take any mail type other than their specialisation. Clearly the first approach is incapable of reducing $\boldsymbol{\ell}.1$, and the second of reducing $\boldsymbol{\ell}.3$. Clearly any "good" algorithm must compromise between greed and selectivity.

3.1. Threshold Based Algorithms

Threshold based algorithms can be seen as an advance on the selective algorithm outlined above. Each agent has a set of thresholds related to its desire to engage in particular tasks. In this case, as tasks types are represented by mail types, each agent a has a set of thresholds $\boldsymbol{\theta}_a = (\theta_{a,1}, .., \theta_{a,N_m})$, where $\theta_{a,m}$ is the agent's threshold for mail type m. This threshold is related to the *stimulus* s of a particular task instance, indicating the demand of a task for completion using a threshold function, Θ which gives the probability of engaging in a task given a stimulus and corresponding threshold. The probability of engagement should be high for $s >> \theta$, low for $s << \theta$, zero for $s = 0$ (no demand for the task), and $\frac{1}{2}$ for $s = \theta$. Thus, for appropriate thresholds, agents will behave selectively in most cases but, for extreme stimulus, are capable of adjusting their behaviour.

Here we have chosen to use the exponential threshold function (ETF), which is the standard threshold function as proposed by Bonabeau et al. [3], and is defined, for threshold θ and stimulus s, as:

$$\Theta(s, \theta) = \begin{cases} \frac{s^\lambda}{s^\lambda + \theta^\lambda} & \text{if } s \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

which for $\lambda \geq 1$ has all the desired properties. In this problem, the stimulus is the waiting time of the batch, a fairly natural measure of its demand for completion. Hence, upon encountering a batch of mail type m, an agent a, the probability for acceptance is given by $\Theta(w_{c,m}, \theta_{a,m})$.

When several agents visit a city, they are permitted to act one at a time in a random order. The acting agent examines each piece of mail, again, one at a time in a random order, while the probability of mail uptake given by its threshold function and thresholds. The acting agent continues until it either has accepted a batch, or has rejected all. It is clear that for given threshold function the action of an agent at a city (and its short term efficiency) critically depends on its thresholds, while its flexibility to adapt to new situations (and therefore its long term efficiency) critically depends on its ability to modify the thresholds.

To adjust their behaviour to the state of the system, agents update their thresholds through self-reinforcement using a strategy determined by an **update rule**. After taking mail type m, an agent replaces its thresholds with $U(\boldsymbol{\theta}, m) = (u(\theta_1, m), ..., u(\theta_{N_m}, m))$, where $u(\theta_m, m) < \theta_m$ and $u(\theta_j, m) > \theta_j$, $\forall\, j \neq m$. Thus, agents increase the chance of taking type m (decreasing $\boldsymbol{\ell}.2$) mail and decrease the chance of taking other mail types (decreasing $\boldsymbol{\ell}.1$). It is clear that a good update rule should drive an agent's thresholds to a state in which θ_{a,σ_a} is very low, thus avoiding $\boldsymbol{\ell}.2$ altogether. In this paper, we consider two threshold-based algorithms.

The **Variable Response Threshold (VRT)** update rule was proposed by Theraulaz et al. [32] and was originally applied to the current problem by Price and Tinǒ [26], Price [25]. The change in threshold $\Delta\theta_m$ over a period of time t, is given by:

$$\Delta\theta_m = -\varepsilon\Delta t_m + \psi(t - \Delta t_m) \tag{4}$$

where Δt_m is the time spent performing task m, where ε, ψ are positive constants, and where θ_m is restricted to the interval $[\theta_{min},\theta_{max}]$. For the current model, (4) can be discretised, taking into account the fact that thresholds are only changed when a task is performed. Therefore, when the update rule is called, over a single time step, t will be 1 and Δt_m is 1 if mail type m was taken and 0 otherwise, such that the VRT rule can be rewritten as

$$u(\theta_m, i) = \begin{cases} \theta_m - \varepsilon & \text{if } i = m, \\ \theta_m + \psi & \text{otherwise.} \end{cases} \tag{5}$$

A drawback of the VRT rule is that, in the event of a changeover, for small ε and ψ agents are unlikely to change their thresholds enough to have a high chance of picking the new mail type in the next time step, thus increasing $\boldsymbol{\ell}.2$ and potentially $\boldsymbol{\ell}.1$. In order to overcome these flaws and to see if better efficiency could be obtained, we introduce another update rule.

The **Switch-Over (SO)** update rule, introduced by Goldingay and van Mourik [13], updates thresholds in a very simple manner: by fully specialising in the most recently taken mail type, and fully de-specialising in all other mail types:

$$u(\theta_m, i) = \begin{cases} \theta_{min} & \text{if } i = m, \\ \theta_{max} & \text{otherwise.} \end{cases} \tag{6}$$

In some sense the switch-over rule can be seen as an extreme case of the VRT rule with ε, $\psi \geq \theta_{max} - \theta_{min}$. Such values, however, are not in the spirit of the VRT, and so it makes sense to consider them as separate cases. We introduce SO as we expect it to minimise $\boldsymbol{\ell}.1$ and $\boldsymbol{\ell}.2$ in a static environment, while a drawback is that $\boldsymbol{\ell}.3$ could be maximised.

For ease of notation, we refer to the VRT (resp. SO) algorithm rather that "a threshold-based algorithm using the VRT (resp. SO) update rule".

3.2. Market Based (MB) Algorithm

The MB algorithm that we use, can be seen as a refinement of the greedy style algorithm described at the beginning of this section. In the market based framework, originally developed for this problem type by Morley [23], the agents at a city are offered batches of available mail in a random order. Each agent must submit a bid, determined by a *bidding function*, for the offered batch of mail, with the highest bidder being assigned the batch. In the case of equal high bids, the winner is selected at random from the highest bidding agents. This means that, if there are sufficient agents available at a city, all tasks must be assigned in a similar fashion to the greedy algorithm. The added selectivity is determined by the agents' bidding function.

We use a bidding function based on that developed by Campos et al. [4]. The reasoning behind this function was to cause an agent a to submit large bids for a task of type m and waiting time w if

1. the task has a high priority (w is high),
2. the task type matches its specialisation ($\sigma_a = m$),
3. the agent can process the batch of mail soon (small backlog in queue),

and is of the form

$$B_a(w, m) = \frac{\omega_p w \times (1 + \omega_s \delta_{m,\sigma_a})}{T_a(m)^{\omega_t}} \tag{7}$$

where δ is the Kronecker delta. ω_p and ω_s weight the bid in relation to the priority of the task for completion and the selectivity (desire to take mail of its specialised type) of the agent respectively. We define

$$T_a(m) = \sum_{l=1}^{L_q} q_{a,l} + \delta_{m,\sigma_a} t_p + (1 - \delta_{m,\sigma_a}) t_c \tag{8}$$

as the time it would take agent a to clear its queue and finish processing a batch of mail of type m, which is exponentially weighted by ω_t. However, as the set of agents is homogeneous and the acceptance of a bid is only determined

relative to other bids, the priority w and weight parameter ω_p can be removed from eq. (7), to yield the bidding function used in this paper:

$$B_a(m) = \frac{1 + \omega_s \delta_{m,\sigma_a}}{T_a(m)^{\omega_t}} \tag{9}$$

Note that the bidding function in eq. (7) takes into account the waiting time of the mail batch, which is not relevant to our objective. However, Campos et al. [4] use this function to maximise efficiency (our objective) and minimise the number of changeovers (closely related to our objective). This is possible because, for the optimal parameter choices used in [4], the relative values of bids are dominated by $T_a(m)$ and δ_{m,σ_a}. The priority of a task (i.e. the waiting time) is in effect only used in a tie-break situation between to otherwise similar bids.

3.3. Hybrid Algorithm

The above algorithms contain only the *essential* features of threshold- and market-based task allocation algorithms (threshold based acceptance and auction based assignment respectively). The selectivity of the VRT and SO algorithms means that they may have higher $\boldsymbol{\ell}.2$ than is necessary. Although the MB algorithm does not suffer from $\boldsymbol{\ell}.2$ it is unable to actively decrease $\boldsymbol{\ell}.1$ by increasing $\boldsymbol{\ell}.3$.

To investigate what (combination of) strategies is optimal in a given situation, we consider a simple hybrid of the VRT and MB algorithms in which all agents have a set of thresholds, but the order of the action of agents is determined in an auction identical to the MB algorithm described above with bidding function (9). After each round of bidding, the winning agent is then offered the mail type it has bid for which it accepts or rejects using the ETF. In the case of rejection, it then looks at the other mail at the city in a random order, accepting or rejecting according to the ETF exactly as it would in a standard threshold-based algorithm. Upon acceptance, it updates its thresholds according to the VRT update rule in eq. (5).

Note that our hybrid algorithm shares similarities with the threshold-based R-WASP algorithm studied by Cicirello and Smith [6, 7], Nouyan et al. [24], and to the market-based algorithm with a "reserve price" studied by Kalra and Martinoli [17]. All three algorithms have the ability to select mail (similar to threshold-based algorithms) and to assign mail to the individual who is best suited to it (similar to the market-based algorithms).

Note also that our focus is on the qualitative behaviour of task allocation strategies and their combinations. As these three algorithms are qualitatively similar, we concentrate on our combination of VRT and MB rules. As it can interpolate between the pure VRT and pure MB algorithms, it allows us to identify the exact combination of strategies responsible for a particular behaviour. When $\theta_{max} = 0$, agents always accept the first batch of mail offered to them (i.e. the one that they have won an auction to be offered) thus making the algorithm purely MB. Similarly, when $\omega_s = \omega_t = 0$, agents all submit identical bids of 1 to the auction, and they act in a random order exactly as in a pure VRT algorithm. For any other parameter choice, the algorithm is truly hybrid. As already noted, the SO algorithm is identical to the VRT algorithms with $\epsilon = \psi = \theta_{max} - \theta_{min}$.

3.4. Particle Swarm Optimisation

As in any given environment, the behaviour of the previous algorithms is governed by a set of parameters, for a fair comparison of the performance of the algorithms it is necessary to ensure that none is unfairly disadvantaged by poor parameter choices. Therefore, we optimise the parameter sets to determine the best performance of each algorithm in each environment.

Genetic algorithms (GAs) have previously been used to optimise parameters in such settings: e.g. Bonabeau et al. [2] have used a GA to cause a stigmergic system to build a structured architecture autonomously (where previously the system required guidance towards particular structures), while Campos et al. [4] have successfully used a GA to optimise a threshold model on a similar problem, to name but a few. Here, we apply another evolutionary algorithm, namely particle swarm optimisation (PSO), to optimise all parameters.

Particle swarm optimisation is a swarm intelligence based algorithm, developed by Kennedy and Eberhart [18], which has emerged as a viable alternative to the GA. Similarly to the GA it comprises a population of individuals (or particles) moving in, and evaluating, the fitness landscape. However, rather than relying on random moves to improve their fitness, these particles retain information about good locations and share them with the rest of the swarm. Particles are attracted to these good locations allowing an efficient search of the fitness space. This often allows PSO to find comparable solutions to naive GAs in fewer function evaluations [11]. The efficiency of PSO, combined with

the relative simplicity of a GA, makes it a good general purpose search heuristic, particularly in engineering contexts when solution speed is an important concern [30].

A PSO swarm is composed of a set of n particles "flying" through some search space $\mathcal{S}$ to minimise a cost function $f : \mathcal{S} \to \mathbb{R}$. At time t a given particle i has a *position*, $\vec{x}_i(t)$ and a *velocity*, $\vec{v}_i(t)$. It can also remember its *historic best* position, $\vec{h}_i(t)$, the lowest cost position it has visited over the course of a run. In addition to remembering good positions, particles socially share information about their historic bests in *neighbourhoods*. This gives them access to a *neighbourhood historic best*, $\hat{h}_i(t)$, the best position found by any particle within their neighbourhood. In order to make our PSO algorithm more robust to local minima we use the neighbourhood defined in the **lbest** version of PSO (Eberhart and Kennedy [10]). In this version, given an enumeration of the particles, the i^{th} particle has a neighbourhood given by $\{i, i \pm 1 \bmod n\}$.

Given these variables, a particle, i then updates its velocity according to the following formula:

$$\vec{v}_i(t) = \eta(t)\vec{v}_i(t-1) + c_1\,\vec{r}_1 \otimes (\vec{h}_i(t) - \vec{x}_i(t)) + c_2\,\vec{r}_2 \otimes (\hat{h}_i(t) - \vec{x}_i(t)) \tag{10}$$

where $\otimes$ is the component wise vector product (i.e. $\vec{a} \otimes \vec{b} = (a_1 b_1, a_2 b_2, ...)$) and c_1 and c_2 are constants determining the relative magnitude of the "cognitive" and "social" parts of the equation respectively. The inertial weight $\eta(t)$, introduced by Shi and Eberhart [31], is taken to be a linearly decreasing function of t, in order to attain good global search at the beginning, and good local search at the end.

As we generally wish to optimise the parameters in some bounded subspace, $\vec{v}_i(t)$ is then clamped so that its magnitude in each dimension does not exceed the size of the subspace. Finally, we update each particle's position using $\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t)$, and move it to the nearest point on the boundary if $\vec{x}_i(t+1)$ falls outside $\mathcal{S}$.

Since we optimise a stochastic system we need to ensure that our algorithm is robust with respect to stochasticity. Pugh et al. [27] have developed a simple variant of PSO in which, if a particle's historic best position is not replaced in a given iteration, its cost is re-evaluated and its true cost is taken to be the average of all previous values. As this variant outperforms standard PSO in the presence of noise, we use it when optimising parameters.

4. Theoretical Analysis

In general it is hard to theoretically describe the performance of the agent-based algorithms on the mail retrieval problem, as they depend on continuous variables (thresholds, bidding weights), such that the number of micro-states of the agents is infinite (not countable). In other contexts where this problem occurs, such as continuous models on sparse random graphs (see e.g. Kühn et al. [20] and references therein), population dynamics can be used for the theoretical analysis. For agent based models, however, this is tantamount to simulating the model. Nevertheless, we can derive mail uptake rule- independent theoretical upper bounds for the efficiency of an infinite population in *ideal circumstances*, i.e. when no mail is lost due to $\boldsymbol{\ell}$.1-$\boldsymbol{\ell}$.3. This situation would occur when $t \leq L_q$ and when agents *never* reject mail such that the efficiency is only limited by $\boldsymbol{\ell}$.4.

Then, both the agent profile and the mail waiting times become irrelevant and the efficiency is a function of the profile of the following simplified city micro-states alone:

$$C = \boldsymbol{b}_{N_m} = (b_1, .., b_{N_m})\,, \tag{11}$$

where $b_i \in \{0, 1\}$ is the availability of mail type i at the city. The set $\mathcal{S}_C$ of all possible states has cardinality $|\mathcal{S}_C| = 2^{N_m}$. Defining the states of the cities as $\boldsymbol{S}(t) = \{S_c(t), c = 1, .., N_c\}$, at any time t the global state of the system is completely determined by the city profile $\boldsymbol{\chi}(t) = \{\chi_C(t),\ C \in \mathcal{S}_C\}$, where

$$\chi_C(t) \equiv \frac{1}{N_c} \sum_{c=1}^{N_c} \delta_{S_c(t),C}\,. \tag{12}$$

In the large system limit, as a consequence of the Central Limit Theorem, $\chi_C(t)$ become deterministic quantities, for which we can derive the exact time evolution. It is convenient to break up this time evolution into two distinct steps:

1) changes to the χ_C during mail uptake.

2) changes to the χ_C during mail production.

The change in city profile during mail uptake can be described by multiplication with a matrix $\mathbf{L}$, while the change in city profile during mail production can be described by multiplication with a matrix $\mathbf{P}(t)$ which is time dependent for the dynamic environment only. Combined, this gives the following exact time evolution for the city profile:

$$\boldsymbol{\chi}(t+1) = \mathbf{P}(t)\,\mathbf{L}\,\boldsymbol{\chi}(t)\,. \tag{13}$$

Then, the efficiency $E(t)$ (the probability that an agent takes mail at time t) is given by

$$E(t) = \sum_{k=1}^{N_m} \chi_k(t)\Big(1 - P_{R_{a/c}}(k) + \frac{k - R_{a/c}}{R_{a/c}} \sum_{j=k+1}^{\infty} P_{R_{a/c}}(j)\Big), \tag{14}$$

where $\chi_k(t) \equiv \sum_{\boldsymbol{b}\in\mathcal{S}_C} \chi_{\boldsymbol{b}}(t)\delta_{|\boldsymbol{b}|,k}$ is the probability that a city has exactly k pieces of mail available, P_λ is the Poisson distribution with parameter λ, and $R_{a/c}$ is the ratio of agents to cities. The details of these derivations and the exact expressions of the matrices $\mathbf{L}$ and $\mathbf{P}(t)$ can be found in the appendix. A comparison between the performance of the various update rule/threshold function combinations with this theoretical upper bound is presented in the following section.

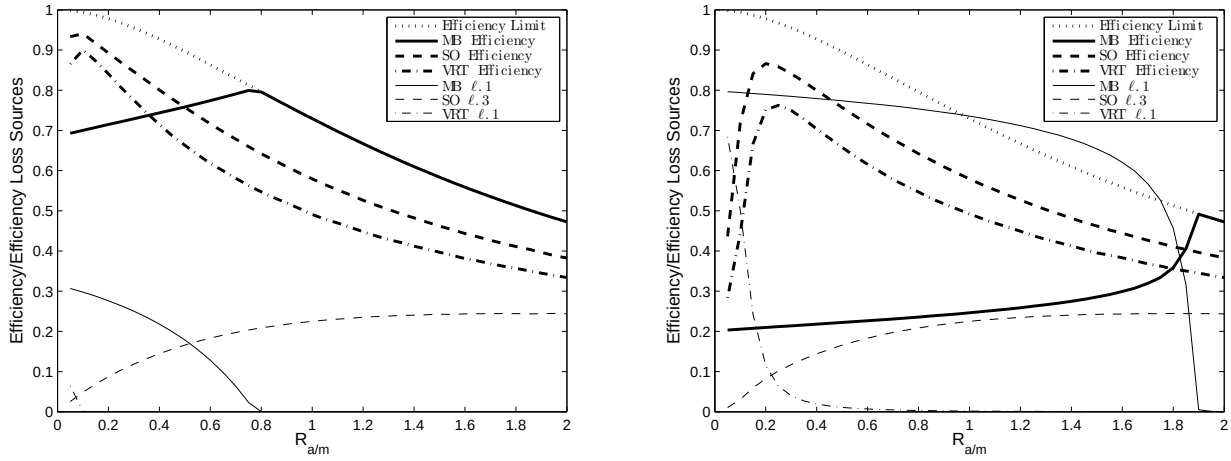
5. Results

In this section, we discuss the numerical results. First, we investigate the influence of the three load measures mentioned in section 2.2, the system parameters $R_{a/m}$, t_c and N_m, on qualitative algorithmic performance. Secondly, we optimise the algorithms' parameters using PSO and compare their absolute performances in terms of overall efficiency.

A full investigation of the effect of varying parameters is beyond the scope of this paper. Therefore we have opted to investigate the influence of load on the efficiency systematically, by varying one load measure at a time and keeping the rest in *the standard setting*: unless specified otherwise we simulate the system with $N_a = 5 \times 10^4$ agents. This has been shown to be large enough to remove finite size effects for threshold-based algorithms [13], and similar results were obtained for the MB algorithms. We also take $N_m = 2$ mail types, a processing time $t_p = 1$ and a queue length $L_q = 10$. Results are given for both low ($t_c = 2$) and high ($t_c = 10$) levels of changeover time. In order to have a fair comparison between different environments, we take $R_{a/m} = 1$ in a static environment, while in the dynamic environment we take $R_{a/m} = 0.5$ (as $\overline{\pi_m(t)} = 0.5$ over a period). We fix the various parameters to the following values: for the threshold-based algorithms, $\theta_{min} = 0$, $\theta_{max} = 50$, $\varepsilon = \psi = 5$ and $\lambda = 2$. For the MB algorithms we take values approximately equal to the values found using a GA by Campos et al. [4], with $\omega_s = 1750$ and $\omega_t = 4$. A standard run consists of 500 iterations over which the average efficiency per agent is monitored, and the standard dynamic environment has a period $\xi = 50$. Note that all simulations are implemented in C++ and are performed on a linux-PC cluster.

The results presented here are intended to be representative of a larger investigation and, as such, we make the following comments. As the parameters we have chosen are by no means guaranteed to be ideal for a given setting, comparing the the algorithms quantitatively is not appropriate (until section 5.2, when we optimise the parameters). Instead we compare the qualitative trends of the algorithm. We present qualitative results only for the static environment, as the dynamic environment has the same trends.

Note that, over the course of our investigation, we observe that changes in efficiency caused by varying a system parameter are mainly triggered by changes in: $\boldsymbol{\ell}.1$ in the case of the MB algorithms; $\boldsymbol{\ell}.3$ in the case of the SO algorithm; $\boldsymbol{\ell}.1$ and $\boldsymbol{\ell}.3$ in the case of the VRT algorithm. However, $\boldsymbol{\ell}.3$ for the VRT algorithm exhibits qualitatively similar behaviour to $\boldsymbol{\ell}.3$ for the SO algorithm in most cases. Therefore, for the sake of clarity, when presenting all three algorithms on the same graph we will only include $\boldsymbol{\ell}.1$ for the MB algorithm and $\boldsymbol{\ell}.3$ for the SO algorithm. For the VRT algorithm we display $\boldsymbol{\ell}.1$ instead of $\boldsymbol{\ell}.3$ except in cases where $\boldsymbol{\ell}.1$ is negligible (figure 2) or where $\boldsymbol{\ell}.3$ exhibits qualitatively different behaviour from SO (figure 3 (a)).



(a) (b)

Figure 1: Efficiency and loss sources as function of $R_{a/m}$ for $N_m = 2$ in the static environment for $t_c = 2$ (a) and $t_c = 10$ (b). For the threshold-based algorithms, and with the exception of very low values of $R_{a/m}$, where $\boldsymbol{\ell}.1$ dominates, the efficiency follows the same trend as the theoretical upper bound. As expected loss $\boldsymbol{\ell}.1$ becomes negligible and $\boldsymbol{\ell}.3$ increases as $R_{a/m}$ increases. The deviation from the theoretical limit is less pronounced for $t_c = 2$. For the the MB algorithm efficiency increases with $R_{a/m}$, coupled with decreasing $\boldsymbol{\ell}.1$, until it reaches the theoretical limit. It then immediately switches to a downward trend taking the same value as the limit.

5.1. Qualitative Trends

In figure 1, we compare the upper bound with the actual efficiency and the loss sources of the algorithms (for $N_m = 2$), as a function of $R_{a/m}$. For threshold-based algorithms the low efficiency at low $R_{a/m}$ is due to high average waiting times which become close enough to θ_{max} to overwhelm agents' selectivity and force multiple changeovers and high levels of $\boldsymbol{\ell}.1$. Note that this is merely a function of our arbitrarily chosen θ_{max}, progressively increasing this value as $R_{a/m}$ decreases would allow us to overcome this lack of selectivity. At high values of $R_{a/m}$ it is clear that θ_{max} is too high as we know that a smaller population could serve the demand, such that a drop in $\boldsymbol{\ell}.1$ would be acceptable in order to decrease $\boldsymbol{\ell}.2$ and $\boldsymbol{\ell}.3$.

For the MB algorithm a high demand upon agents (i.e. low $R_{a/m}$) leads to $\boldsymbol{\ell}.1$. Therefore, as $R_{a/m}$ increases $\boldsymbol{\ell}.1$ decreases, leading to increasing efficiency until $\boldsymbol{\ell}.1$ reaches a negligible value. At this point $\boldsymbol{\ell}.1$-$\boldsymbol{\ell}.3$ are all negligible satisfying the conditions used to derive the theoretical limit. Thus, the algorithm's efficiency increases until it hits this limit after which it saturates the theoretical limit as $\boldsymbol{\ell}.4$ entirely determines its behaviour. While the increased changeover penalty from figure 1 (a) to (b) makes only a small difference in the demand that threshold-based algorithms can cope with, it renders the MB algorithm unproductive up to a relatively high value of $R_{a/m}$, beyond which $\boldsymbol{\ell}.1$ drops sharply due to its self reinforcing nature.

In figure 2, we observe that the level of t_c has little effect on the threshold-based algorithms. These algorithms tend to reject mail which is of a different type to their effective specialisation, and the penalty caused by high t_c is incurred infrequently, leaving the agents with time to clear their queue backlog. The performance of the MB algorithm, however, is highly determined by t_c. As the algorithm is incapable of rejecting mail offered by a city, it incurs a far higher proportion of changeovers than the threshold-based algorithms. This leads to a huge loss in performance as $\boldsymbol{\ell}.1$, the proportion of agents inactive due to a full queue, increases with t_c.

Figure 3 shows the efficiency as a function of N_m. For the threshold based algorithms, the efficiency initially increases with N_m due to a more uniform distribution of agents over cities, decreasing the likelihood of $\boldsymbol{\ell}.4$. For the SO algorithm the efficiency then levels off, while for the VRT algorithm it decreases as N_m further increases.

For $t_c = 2$ this is caused by an increase in $\boldsymbol{\ell}.3$. At low levels of N_m this matches an increase in $\boldsymbol{\ell}.3$ for the SO algorithm and can be explained by a higher probability of encountering other mail types (i.e. a trade-off with $\boldsymbol{\ell}.4$). The additional increase on top of SO's $\boldsymbol{\ell}.3$ is explained by VRT's lower ability to re-specialise. VRT agents rely on repeated exposure to high stimulus batches of mail of a particular type to force them to change their thresholds

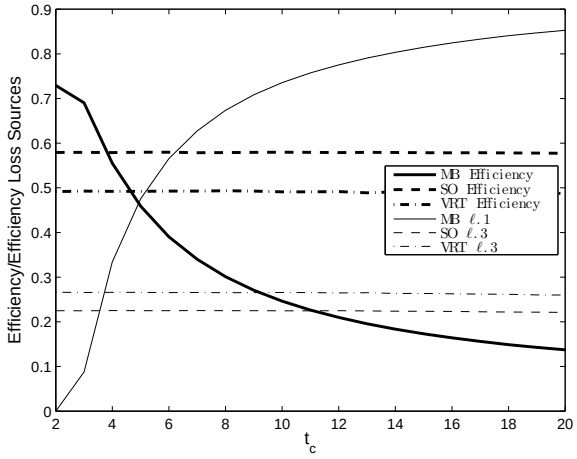
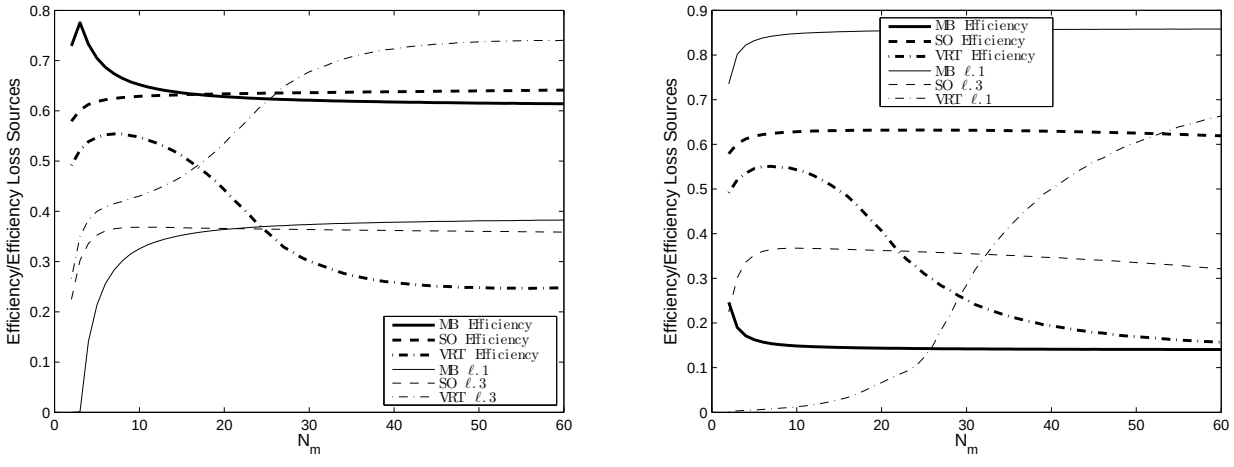


Figure 2: Efficiency and loss sources as function of the changeover time t_c in the static environment. Increasing t_c causes $\boldsymbol{\ell}.1$ to increase for MB algorithms, with a corresponding drop in efficiency. The value of t_c has little effect on the threshold-based algorithms, its only noticeable result being a small increase in $\boldsymbol{\ell}.1$ for the VRT algorithm while other values remain approximately static.



(a) (b)

Figure 3: Efficiency and loss sources as function of N_m in the static for $t_c = 2$ (a) and $t_c = 10$ (b). In both situations the threshold-based algorithms show an initial increase in efficiency, while the MB algorithm shows a sharp increase followed by a decrease, as $\boldsymbol{\ell}.1$ increases, for $t_c = 2$ and a sharp decrease for $t_c = 10$. Both the MB the SO algorithm quickly tend to a relatively stable efficiency, although market exhibits a small downward trend driven by increasing $\boldsymbol{\ell}.3$ and SO a small upward trend driven by decreasing $\boldsymbol{\ell}.3$. After the initial increase the VRT algorithm decreases in efficiency, before tending towards a stable value. For $t_c = 2$ this is caused by $\boldsymbol{\ell}.3$ while for $t_c = 10$ this is caused by an increase in $\boldsymbol{\ell}.1$ (note the change in between (a) and (b)).

to match their specialisation. When N_m is high (and, therefore N_c is low for fixed $R_{a/m}$) there is little chance that a particular agent will repeatedly see a batch of the same under-served mail type. This leads to low pressure to develop an optimal split in specialisations (N_a/N_m agents specialised in each mail type) and an increased chance for similarly specialised agents to visit the same city (high $\boldsymbol{\ell}.3$).

For $t_c = 10$, VRT's loss of efficiency is due to an increase in $\boldsymbol{\ell}.1$. For high N_m, agents must examine more mail before they find their type leading to increased chances of changeovers. As VRT agents are slow to change their thresholds to match their specialisation, this is likely to lead to repeated changeovers and thus, when t_c is high, increased $\boldsymbol{\ell}.1$.

The MB algorithm experiences an initial increase in efficiency for $t_c = 2$, for the same reason as the threshold-based algorithms: a decreased chance of $\boldsymbol{\ell}.4$ due to the increasingly uniform numbers of agents at cities. The following increase in $\boldsymbol{\ell}.1$, and levelling off of efficiency, is caused by agents being forced to bid upon more mail before they are offered their specialised type. For $t_c = 10$, the MB algorithm shows an initial increase in $\boldsymbol{\ell}.1$, merely due to the fact that an increasing fraction of available mail will not be of the specialised type. The efficiency loss then levels off because $\boldsymbol{\ell}.1$ approaches 0.9, the total fraction of $\boldsymbol{\ell}.1$ that would occur if agents always switched mail types with $t_c = 10$. This limit is never reached as market agents don't always have the opportunity to take mail.

While it is important to note which parameter values cause a breakdown in the algorithms, behavioural comparisons are more relevant under *normal* conditions. As high N_m causes a breakdown in the VRT algorithm and, for appropriate t_c, leads to relatively similar behaviour in the SO and the MB algorithms, the fixing of $N_m = 2$ is justified.

To conclude, we observe that both strategies have have different strengths. The selectivity of the threshold-based algorithms (particularly SO) allow them to cope reasonably well with a much greater range of loads. The MB algorithm's task assignment strategy allows it to saturate the bound in conditions of low load, however its lack of selectivity causes its complete break down under high load.

5.2. Performance

Although a full investigation of the effect of all parameters of the algorithms is beyond the scope of this paper, we do wish to ensure that the behaviour observed in the previous section holds for good parameter choices. Therefore, we employ a robust PSO algorithm to find optimised parameters with good performance. We cannot guarantee the parameters to be truly optimal, but they do allow each algorithm to perform well, unhampered by initial parameter choices.

When optimising a given algorithm we use PSO with 20 particles, each having a full candidate set of the variables used by the algorithm. Each set of variables is tested by assigning them to all agents in a smaller scale run (500 iterations with 10^2 agents, which is sufficient to avoid finite size effects, see [13]) of the mail processing problem. The particle's performance is taken to be the average efficiency of this run, and PSO is terminated after $5 \cdot 10^4$ such runs are performed. The final efficiencies presented are calculated using the thus obtained parameters on a system of $5 \cdot 10^4$ agents. The initial parameters used in the PSO algorithm are set according to the "constriction coefficient" method, introduced by Clerc and Kennedy [8], giving $\eta(0) \approx 0.7298$ and $c_1 = c_2 \approx 1.496$, after which the inertial weight is decreased linearly with the number of function evaluations $f_n(t)$:

$$\eta(t) = \eta(0)\left(1 - \frac{f_n(t)}{f_{max}}\right) \tag{15}$$

where $f_{max} = 5 \cdot 10^4$ is the maximum number of evaluations.

As an exhaustive investigation of the optimised performance of these algorithms under all possible combinations of system parameters (or even under individual variation of each system parameter) would be impractical, we choose to test the algorithms' performances in four representative settings: the static and dynamic settings with either $t_c = 2$ or $t_c = 10$. The static and dynamic settings are representative of an algorithms' performance when stability, respectively adaptability is required. Similarly, $t_c = 2,\ 10$ are representative for low and high load situations respectively. Under low load, immediate task completion is more important than the short-term performance of individual agents, while high load requires selectivity to avoid a fatal cascade of agent failures.

We optimise the performance of three different algorithms: the SO and MB algorithms (as they outperform VRT in all circumstances), and the hybrid VRT algorithm. It allows us to test whether the hybrid algorithm can outperform the individual algorithms on which it is based. Given its ability to evolve arbitrarily close to any of the other algorithms

Table 1: Final efficiencies (as proportions of the theoretical upper limit) with $t_c = 2$

Environment	Static	Dynamic
Original VRT	0.492156 (0.674807)	0.406784 (0.622904)
Original SO	0.579163 (0.794104)	0.463681 (0.710030)
Original MB	0.729365 (1.00004)	0.653368 (1.00050)
Optimised SO	0.704393 (0.965810)	0.653255 (1.00032)
Optimised MB	0.729369 (1.00005)	0.653368 (1.00050)
Optimised VRT Hybrid	0.72926 (0.999905)	0.653203 (1.000243)

Table 2: Final efficiencies (as proportions of the theoretical upper limit) with $t_c = 10$

Environment	Static	Dynamic
Original VRT	0.491360 (0.673715)	0.405367 (0.620735)
Original SO	0.579069 (0.793975)	0.463695 (0.710052)
Original MB	0.246390 (0.337831)	0.250129 (0.383020)
Optimised SO	0.630432 (0.864400)	0.513024 (0.785589)
Optimised MB	0.246399 (0.337843)	0.249985 (0.382800)
Optimised VRT Hybrid	0.645127 (0.884549)	0.536984 (0.822278)

(including the VRT algorithm), it gives us a method to determine the optimal single algorithm. Every algorithm was optimised for each setting it was tested in (high and low load, static and dynamic environments). Note that PSO provided no improvement for the MB algorithm over the standard settings, due to its relative insensitivity to parameter choice [19].

Table 1 illustrates the efficiency of the various algorithms for $t_c = 2$. These efficiencies are given in absolute numbers and as a fraction of the theoretical upper bound (in brackets). Note that fractions of the theoretical limit > 1 are possible as our system, though large, is still stochastic.

We can see that even the original MB algorithm can reach the performance limit in this low load setting. The hybrid algorithm reaches the same level of performance by essentially becoming purely MB, finding optimised parameters with $\theta_{max} \approx 0$. This shows that in low load settings selectivity is not an important property, as backlogs in an agent's queue have time to clear so agents rarely become inactive as a result of changeovers. The parameters of the optimised SO algorithm ($\theta_{max} < 1$ for both the static and dynamic environments) further support this claim.

Table 2 gives the efficiency of our algorithms for $t_c = 10$. In this situation with high load, the efficiency of the MB approach falls far below that of the threshold-based algorithms due to its inherent inability to reject mail. The selectivity of the SO algorithm allows it to perform efficiently in this high load setting. The hybrid algorithm slightly outperforms SO by essentially using a SO based threshold mail selection/rejection method, while using a MB approach to increase its chance of seeing its desired mail type if possible.

To conclude, we observe that in a low load setting selectivity is not required and so the MB algorithm is optimal. When load is high, selectivity is the most important feature meaning that SO is a close to optimal strategy. However, a hybrid strategy can match the low load performance of the MB algorithm while maintaining the robustness to high load of the threshold-based algorithms.

6. Conclusions and Outlook

In this paper, we have studied both nature inspired and market-based algorithms for distributed task allocation applied to a problem of mail processing. We have investigated both the algorithms' ability to cope with load, and their

adaptability. In particular, we found that nature inspired, threshold-based algorithms have a far higher robustness to high load than the MB algorithm although the MB algorithm is optimal in low load settings.

We have identified the various loss sources, and have demonstrated that the random choice of cities to visit by the agents forms the main limitation on the maximal attainable efficiency. We have derived this limit theoretically. We have also investigated the absolute absolute performance of the algorithms in relation to this limit and have therefore introduced a new hybrid approach. For a fair comparison, we have used a particle swarm optimisation algorithm to find good parameter sets for all algorithms. Our hybrid algorithm has allowed us to identify optimal combined strategies in both low and high load regimes, matching the efficiency of the MB algorithm in a low load setting by becoming purely market based. In a high load setting, we identify an optimal combination of strategies that outperforms the optimised versions either single algorithm.

While our hybrid algorithm is capable of adopting good strategies for both high and low load settings, it requires a separate adjustment of parameters. This needs to be addressed for it to be fully effective. Also, the performance of our algorithms may still be limited by our choice of the functional forms of the algorithms and bidding functions. As such it would be interesting to apply genetic programming in which agents are allowed to develop their strategies freely.

Acknowledgements

The authors would like to thank P. Tiňo, A. Ekart and J. P. Neirotti for stimulating discussions, useful suggestions, and careful reading of the manuscript.

[1] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm intelligence: from natural to artificial systems, Oxford University Press, Inc., New York, NY, USA, 1999.

[2] E. Bonabeau, S. Guerin, D. Snyers, P. Kuntz, G. Theraulaz, Three-dimensional architectures grown by simple 'stigmergic' agents, biosystems 56 (2000) 13–32.

[3] E. Bonabeau, G. Theraulaz, J.L. Deneubourg, Fixed response thresholds and the regulation of division of labor in insect societies, Bulletin of Mathematical Biology 60 (1998) 753–807–807.

[4] M. Campos, E. Bonabeau, G. Theraulaz, J.L. Deneubourg, Dynamic scheduling and division of labor in social insects, Adaptive Behavior 8 (2000) 83–95.

[5] Y. Chevaleyre, P.E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J.A. Rodríguez-aguilar, P. Sousa, Issues in multiagent resource allocation, Informatica 30 (2006) 2006.

[6] V. Cicirello, S. Smith, Distributed coordination of resources via wasp-like agents, in: W. Truszkowski, C. Rouff, M. Hinchey (Eds.), Innovative Concepts for Agent-Based Systems, pp. 71–80.

[7] V.A. Cicirello, S.F. Smith, Wasp-like agents for distributed factory coordination, Journal of Autonomous Agents and Multi-Agent Systems 8 (2004) 237–266.

[8] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, Evolutionary Computation, IEEE Transactions on 6 (2002) 58–73.

[9] M.B. Dias, R. Zlot, N. Kalra, A. Stentz, Market-based multirobot coordination: A survey and analysis, Proceedings of the IEEE 94 (2006) 1257–1270.

[10] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS '95., pp. 39–43.

[11] R. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in: V. Porto, N. Saravanan, D. Waagen, A. Eiben (Eds.), Evolutionary Programming VII, volume 1447 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 1998, pp. 611–616. 10.1007/BFb0040812.

[12] H. Goldingay, J. van Mourik, The influence of memory in a threshold model for distributed task assignment, in: S.A. Brueckner, P. Robertson, U. Bellur (Eds.), SASO '08: Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, IEEE Computer Society, Washington, DC, USA, 2008, pp. 117–126.

[13] H. Goldingay, J. van Mourik, Evolution of competing strategies in a threshold model for task allocation, in: R. Lee, J. Ma, L. Bacon, W. Du, M. Petridis (Eds.), Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2010, volume 295 of *Studies in Computational Intelligence*, Springer Berlin / Heidelberg, 2010, pp. 85–98.

[14] M. Hoeing, P. Dasgupta, P. Petrov, S. O'Hara, Auction-based multi-robot task allocation in comstar, in: E.H. Durfee, M. Yokoo, M.N. Huhns, O. Shehory (Eds.), AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, ACM, 2007, pp. 1–8.

[15] B. Hong, V.K. Prasanna, Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput, Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, in: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, pp. 52+.

[16] P. Janacik, T. Heimfarth, F. Rammig, Emergent topology control based on division of labour in ants, in: AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications, IEEE Computer Society, Washington, DC, USA, 2006, pp. 733–740.

[17] N. Kalra, A. Martinoli, Comparative study of market-based and threshold-based task allocation, in: M. Gini, R. Voyles (Eds.), Distributed Autonomous Robotic Systems 7, Springer Japan, 2007, pp. 91–101.

[18] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings., IEEE International Conference on, volume 4, pp. 1942–1948.
[19] O. Kittithreerapronchai, C. Anderson, Do ants paint trucks better than chickens? markets versus response thresholds for distributed dynamic scheduling, in: Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, volume 2, pp. 1431–1439.
[20] R. Kühn, J. van Mourik, M. Weigt, A. Zippelius, Finitely coordinated models for low-temperature phases of amorphous systems, Journal of Physics A: Mathematical and Theoretical 40 (2007) 9227–9252.
[21] K. Lerman, C. Jones, A. Galstyan, J. Maja, Analysis of dynamic task allocation in multi-robot systems, International Journal of Robotics Research 25 (2006) 225–242.
[22] K.H. Low, W.K. Leow, M.H. Ang, Task allocation via self-organizing swarm coalitions in distributed mobile sensor network, in: D.L. McGuinness, G. Ferguson (Eds.), AAAI'04: Proceedings of the 19th national conference on Artifical intelligence, AAAI Press, 2004, pp. 28–33.
[23] D. Morley, Painting trucks at general motors: The effectiveness of a complexity-based approach, Embracing Complexity: Exploring the Application of Complex Adaptive Systems to Business (1996) 53–58.
[24] S. Nouyan, R. Ghizzioli, M. Birattari, M. Dorigo, An ant-based algorithm for the dynamic task allocation problem, Technical Report TR/IRIDIA/2004-16, IRIDIA, Université Libre de Bruxelles, 2004.
[25] R. Price, Evaluaton of adaptive nature inspired task allocation against alternate decentralised multiagent strategies, BSc Dissertation, University of Birmingham, 2004.
[26] R. Price, P. Tinŏ, Evaluaton of adaptive nature inspired task allocation against alternate decentralised multiagent strategies, in: Parallel Problem Solving from Nature VIII, Springer Berlin / Heidelberg, 2004, pp. 982–990.
[27] J. Pugh, Y. Zhang, A. Martinoli, Particle swarm optimization for unsupervised robotic learning, in: Swarm Intelligence Symposium, pp. 92–99.
[28] M. Saleem, G.A.D. Caro, M. Farooq, Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions, Information Sciences In Press, Corrected Proof (2010) –.
[29] M.C. Schut, On model design for simulation of collective intelligence, Information Sciences 180 (2010) 132 – 155. Special Issue on Collective Intelligence.
[30] N. Sharma, A.K. Tarcar, V.A. Thomas, K. Anupama, On the use of particle swarm optimization for adaptive resource allocation in orthogonal frequency division multiple access systems with proportional rate constraints, Information Sciences In Press, Corrected Proof (2011) –.
[31] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pp. 69–73.
[32] G. Theraulaz, E. Bonabeau, J.L. Deneubourg, Response threshold reinforcements and division of labour in insect societies, Proceedings of the Royal Society B: Biological Sciences 265 (1998) 327–332.
[33] W. Xiang, H.P. Lee, Ant colony intelligence in multi-agent dynamic manufacturing scheduling, Eng. Appl. Artif. Intell. 21 (2008) 73–85.

Appendix A. Theoretical Upper Bound for the Efficiency

We derive the upper bound for the efficiency in *ideal circumstances*, when no mail is lost due to $\boldsymbol{\ell}$.1-$\boldsymbol{\ell}$.3, and the efficiency is only limited by $\boldsymbol{\ell}$.4. Then all agents have an identical set of thresholds $\boldsymbol{\theta} = \{\theta_n (= \theta_{min} = 0), \forall n = 1..N_m\}$. Mail uptake is independent of the waiting time, and only depends on the availability of the different mail types. Hence, the city states can be simplified to $C = \boldsymbol{b} \in \{0,1\}^{N_m}$ where $b_m = 1$ when the mail type m is available, and $b_m = 0$ when it is not.

Agents visit cities randomly such that the probability that a subset of i agents visits any given city, is given by

$$\binom{N_a}{i}\left(\frac{1}{N_c}\right)^i\left(\frac{N_c-1}{N_c}\right)^{N_a-i} \simeq P_{R_{a/c}}(i), \tag{A.1}$$

where $R_{a/c} \equiv N_a/N_c$ is the ratio of agents to cities, and P_λ is the Poisson distribution with parameter λ. Since agents are indistinguishable, the probability that an agent takes mail when visiting a city with k available mail types and i visiting agents, is given by:

$$U(k,i) = \begin{cases} \frac{k}{i}\,, & \text{if } i > k\,, \\ 1\,, & \text{if } i \le k\,. \end{cases} \tag{A.2}$$

The total probability that an agent takes mail (i.e. the efficiency) is thus:

$$E(t) = \sum_{k=1}^{N_m}\chi_k(t)\sum_j P_{R_{a/c}}(j-1)\,U(k,j) = \tag{A.3}$$
$$\sum_{k=1}^{N_m}\chi_k(t)\left(1 - P_{R_{a/c}}(k) + \frac{k-R_{a/c}}{R_{a/c}}\left(1-\sum_{j=0}^{k}P_{R_{a/c}}(j)\right)\right).$$

where $\chi_k(t) \equiv \sum_{\boldsymbol{b}\in\mathcal{S}_C} \chi_{\boldsymbol{b}}(t)\delta_{|\boldsymbol{b}|,k}$ is the probability that a city has exactly k pieces of mail available.

We now derive the expressions of the matrices $\mathbf{P}(t)$ and $\mathbf{L}$, that govern exact time evolution for the city profile:

$$\boldsymbol{\chi}(t+1) = \mathbf{P}(t)\,\mathbf{L}\,\boldsymbol{\chi}(t)\,. \tag{A.4}$$

1. Evolution of city states during mail uptake

We can now write down the elements of the transition matrix $\mathbf{L}$ describing (the probability of) a transition from an city state $\boldsymbol{b}$ to a state $\boldsymbol{b}'$ during the mail uptake stage. It is clear that

$$L_{\boldsymbol{b}',\boldsymbol{b}} = \sum_{i=0}^{\infty} P_{R_{a/c}}(i)\, L_{\boldsymbol{b}',\boldsymbol{b}}(i) \tag{A.5}$$

where $L_{\boldsymbol{b}',\boldsymbol{b}}(i)$ is the corresponding transition probability for a city visited by exactly i agents. Since all agents take mail when available, and since during the uptake phase all mail types are equivalent, we have that

$$L_{\boldsymbol{b}',\boldsymbol{b}}(i) = \begin{cases} \delta_{\boldsymbol{b}',\boldsymbol{0}}, & i \geq |\boldsymbol{b}| \\ \binom{|\boldsymbol{b}|}{i}^{-1} \delta_{|\boldsymbol{b}'|,|\boldsymbol{b}|-i} \prod_{m=1}^{N_m} \left(1 - \delta_{b'_m,1}\delta_{b_m,0}\right), & i < |\boldsymbol{b}| \end{cases} \tag{A.6}$$

In the first case, there are enough agents and all mail is taken. In the second case a random subset of i out of $|\boldsymbol{b}|$ mail types is taken ($\binom{|\boldsymbol{b}|}{i}$ possibilities), such that there are $|\boldsymbol{b}| - i$ mail types left, and these must be of the types that were originally present.

2. Evolution of city states during mail production

The element of transition matrix $\mathbf{P}$ describing (the probability of) a transition from an city state $\boldsymbol{b}$ to a state $\boldsymbol{b}'$ during the mail production stage is relatively straightforward to write down directly:

$$P_{\boldsymbol{b}',\boldsymbol{b}} = \prod_{m=1}^{N_m} \left(\delta_{b_m,1}\,\delta_{b'_m,1} + \delta_{b_m,0}\,(\pi_m(t)\,\delta_{b'_m,1} + (1-\pi_m(t))\,\delta_{b'_m,0})\right), \tag{A.7}$$

where the $\pi_m(t)$ are time dependent for the dynamic environment only.