Optimising task allocation to balance business goals and worker well-being for financial service workforces

Christopher Duckworth[a], Zlatko Zlatev[a], James Sciberras[b], Peter Hallett[b], Enrico Gerding[a]

[a]*School of Electronics and Computer Science, University of Southampton, Southampton, UK*
[b]*OnCorps, Bristol, UK*

Abstract

Purpose

Financial service companies manage huge volumes of data which requires timely error identification and resolution. The associated tasks to resolve these errors frequently put financial analyst workforces under significant pressure leading to resourcing challenges and increased business risk. To address this challenge, we introduce a formal task allocation model which considers both business orientated goals and analyst well-being.

Methodology

We use a Genetic Algorithm (GA) to optimise our formal model to allocate and schedule tasks to analysts. The proposed solution is able to allocate tasks to analysts with appropriate skills and experience, while taking into account staff well-being objectives.

Findings

We demonstrate our GA model outperforms baseline heuristics, current working practice, and is applicable to a range of single and multi-objective real-world scenarios. We discuss the potential for metaheuristics (such as GAs) to efficiently find sufficiently good allocations which can provide recommendations for financial service managers in-the-loop.

Preprint submitted to Journal of Modelling in Management (accepted) *June 6, 2025*

Originality

A key gap in existing allocation and scheduling models, is fully considering worker well-being. This paper presents an allocation model which explicitly optimises for well-being while still improving on current working practice for efficiency.

Keywords: workforce, task allocation, genetic algorithm, financial services

1. Introduction

Financial service companies acquire huge volumes of data which requires timely error checking and resolution. Specifically, asset managers must accurately publish the per-share value (Net Asset Value; NAV) of mutual or exchange-traded funds at the end of each trading day. Publication of the NAV is reliant on accurate trading data with errors leading to misreporting, significant fines, and, ultimately, large-scale commercial impacts for asset managers [1, 2]. Error identification and resolution is often a time consuming and overly manual set of tasks. As data volume increases, companies are finding it increasingly hard to monitor all data streams effectively, placing their workforces under increasing pressure.

The human-led checking, analysis, and correction of large volumes of data has not only increased business risk but has had a detrimental impact on staff well-being. Financial Analysts report unsustainably long workhours [3] and detrimental impacts on physical and mental health [4, 5], which have led to voluntary staff turnover rates rising over the last decade to be over 20% annually [6]. Automation and optimisation of this process is critical to ensure employers meet their legal duty of care to their workers, and ensure the likelihood of errors leading to significant financial and reputational loss is minimised.

One area of possible optimisation is considering how effectively such error checking tasks are allocated across a workforce of analysts. Automatic task allocation can be implemented by leveraging Artificial Intelligence (AI) to find solutions which allocate tasks to analysts with appropriate skills and experience (hence reducing completion time and risk of incompletion). For example, novel modelling techniques have been applied to personnel scheduling [7, 8], task and team assignment for up-skilling workforces [9, 10], selection policies to promote cross-learning to increase throughput [11], and profiling worker skills to maximise efficiency in job assignment [12, 13]. While

these approaches often address efficiency, which in-turn may reduce the average analyst's workhours, they do not sufficiently capture medium-term staff well-being objectives. Finding a solution should be multi-faceted; aiming to balance business orientated goals with those of the worker.

A key gap in existing allocation and scheduling models, is therefore explicitly optimising worker well-being. Well-being objectives range from workload to task satisfaction, with the latter ensuring each analyst receives a fair share of rewarding tasks. Well-being objectives also include the autonomy of the worker and their individual preference for types of task they may like to complete. For financial service sector workforces, where high workload is often unavoidable, a critical aspect is to ensure that analysts continue to feel engaged and rewarded from their task set [6]. Business orientated factors such as ensuring high priority tasks (i.e., those which are particularly time-dependent or most impactful to the NAV) are completed and maximising efficiency across the workforce, may often come at the cost of individual worker well-being factors such as task satisfaction. Financial service sector companies must strike a difficult balance between reducing risk from accurate NAV reporting, while maximising well-being objectives to keep voluntary staff turnover as low as possible. In addition to the disruption of training new workers, the cost of an analyst's job turning over is estimated at 50-125% of the employee's annual salary highlighting the need for optimised well-being, even from a purely economic perspective [14].

In this work, we formulate the task allocation and scheduling procedure as a multi-objective problem to quantify and balance both the needs of worker well-being and business orientated goals of efficiency and throughput. Using historical data, we compute expected task completion times by analyst to quantify worker suitability for an allocation based on skill. By considering this in relation to worker availability, we estimate the likelihood of each analyst completing their allocated tasks to assess business risk and the individual's workload. We characterise well-being based on both the likelihood of the allocated task being a true error (i.e., being a rewarding task), and the preference of the analyst (i.e., which tasks they find most satisfying). Pareto optimisation can then be found by our formal model which evaluates the fitness of a given allocation independently for each worker using these parameterisations of completion likelihood of allocated tasks, and well-being.

In this context, finding sufficiently good solutions efficiently is critical. Implemented as a decision-support system for a human-in-the-loop, allocation solutions must be generated quickly for workforce managers who may

then tweak allocation suggestions. Further, allocations may be re-evaluated as task lists and operational circumstances change through the day, again requiring efficiency. As a result, this problem can be framed as combinatorial optimisation where an exhaustive search (i.e., finding global optimal solution) is not plausible or necessarily required (given that managers may alter the recommended allocation). Therefore, specialised algorithms which can quickly rule out large parts of the search space, such as metaheuristics, are favoured.

One popular choice of metaheuristic, is the Genetic Algorithm (GA) which takes inspiration from evolutionary processes, such as mutation, crossover, and selection. GAs are a global optimisation technique, which utilise a population of candidate solutions across the search space. Evolutionary algorithms are popular choices for extensive multi-objective optimization problems as the set of solutions can explore the pareto front. For the purpose of our formal model, we utilise and validate a GA optimiser [15] which aims to maximise the aggregated multi-objective fitness (or utility) across all analysts to ensure business orientated and worker well-being are balanced.

The key contributions of our work are to:

1. Develop a formal model with probabilistic objective functions (Section 3.2) for task completion probability and well-being (reward, preference). Our objectives optimise to ensure the completion of the highest importance tasks are prioritised in the allocation process. The model also contains heuristics which natively deal with pre-allocated and partially completed tasks, enabling the model to re-evaluate the solution as frequently as required, and emulate real-world scenarios. Our model natively handles different work schedules (e.g., part-time, full-time) for analysts.
2. Validate our formal model for the allocation and scheduling of a set tasks across a workforce. We validate the performance of the GA optimisation relative to baseline heuristics (Figure 2) and perform testing to provide suitable hyperparameters for realistic use cases (Section 4.1).
3. Consider the performance of our formal model in relation to current working practice (Figure 4). We simulate allocations from workforce managers and quantify the added value of of our formal model supported by a metaheurtic such as a Genetic Algorithm (Section 4.5).
4. Evaluate the complexity scaling of single and multi-objectives problems for our formal model through asymptotic analysis. We further perform

empirical analysis to understand the scaling of our formal model in typical real-world scenarios (Figure 3).

We validate our model using task and analyst data from a global top 10 asset manager taken over a two year period from 2020 to 2022. Due to data privacy, we present work here using simulated data drawn from typical working circumstances at the asset manager. In Section 2 we provide further problem context about workforce planning, requirement for explicit optimisation of well-being, and metaheuristics. In Section 3 we introduce the mathematical formulation of our allocation model and simulated data, before validating our models on a variety of test scenarios (Section 4), comparing to baseline heuristics and current working practice. Finally we discuss the human-in-the-loop implementation of our model in Section 5 before concluding in Section 6.

2. Literature Review

Increasing interest in automated approaches to workforce planning has provided a number of novel optimisation techniques aiming to improve allocation and scheduling of both work and workers. Reviews on personnel scheduling have defined taxonomies of areas within staff scheduling and rostering, often focused on increasing efficiency [7, 8]. Even approaches that aim to improve the skills of their workforce through task and team assignment [9, 10], selection policies to promote cross-learning [16, 11] and profiling worker skills [12, 13] still optimise with long-term efficiency in mind. While up-skilling ensures workers develop sufficient skills to progress in their job and cross-learning ensures the overall workforce is better balanced, these approaches do not fully capture the objectives of worker satisfaction. Of particular relevance, we note [17] who develop a multi-objective formal model to allocate jobs in the construction industry optimising for both efficiency and career development.

Business orientated goals of efficiency or throughput may come in direct competition with objectives to maximise worker well-being. Balancing multiple completing objectives has been previously studied in the context of scheduling [18, 19]. Despite this, a current gap in knowledge are systematic task allocation methods which effectively balance the competing objectives of efficiency and well-being. Part of the problem could be due to difficulties in effectively modelling well-being. Contemporary theories of resource planning recognise that the alignment of needs and priorities of the business with

those of those workers is critical for continued productivity [20, 21]. Despite this, even popular theories (e.g., job demand-resource model) only focus on relative workload and resourcing to define worker well-being [17].

Workforce planning is particularly challenging in the financial service sector due to the frequency of over-burden (over 40 hours as standard with junior analysts at Goldman Sachs reporting an average of 95 hours worked per week) [4, 3]. Moreover, human resource management practices in finance are in stark contrast to contemporary theory. Analysts report significant impact to their physical and mental health due to working practices, with >75% reporting work impacting their relationships with friends and family [4, 5]. This practice does not recognise the two-way psychological relationship between employer and employee, championed by contemporary theory, outside of the legal and formal framework that an employee simply obtains work for fiscal reward [22]. Developing working practices (e.g., task allocation which optimises well-being) that better the 'pyschological contract' between employee and employer enables workers to view their relationship as a two-way transaction, rather than one imposed against their interests [20].

To enable workforce planning which can optimise both for business orientated and well-being goals, it is critical to have an approach capable of finding solutions to extensive problems with multiple conflicting objectives or criteria. One option, Multiple-criteria decision making (MCDM), is a structured approach used to explicitly evaluate multiple conflicting criteria. MCDM typically takes a multi-step approach consisting of (i) problem formulation, (ii) objective definition, (iii) criteria selection, (iv) setting alternatives, (v) weighting the criteria and (vi) selecting an appropriate MCDM method. Approaches (collectively often referred to as multi-attribute decision making) include pair-wise comparison (e.g., Analytic hierachy process and analytic network process) [23, 24], distance-based methods (e.g., TOPSIS [25]), outranking methods (e.g., PROMETHEE [26]) and value/utility function methods [27]. Despite success in resource allocation (e.g., land, water [28, 29]), problem formulation typically requires predetermined criteria weighting leading to a single solution. Another option is Reinforcement learning, where an agent learns policies to maximise the 'reward function' or other user-provided reinforcement signals that accumulate from feedback in a dynamic environment. Here, the reward function would look to maximise scheduling efficiency while balancing user-provided input on task and work satisfaction. Despite popularity in the context of dynamic scheduling (see e.g., [30]), continuous feedback on well-being is challenging.

Alternatively, multi-objective optimisation concerns optimising a set of objective functions simultaneously. Despite the positive characteristics of MCDM (e.g., tranparency of decision rules and more intuitive interpretation of outcomes) and reinforcement learning (e.g., adaptability and improvement over time), multi-objective optimisation does not require predetermined weights of each criterion leading to a single solution or continuous feedback on well-being. Instead, sets of solutions are generated in order to explore along a Pareto frontier enabling more detailed understanding between the trade-offs of conflicting objectives.

Given the operational bounds of our proposed model (i.e., requirement to be computationally efficient and only needing to find 'good enough' solutions for a static problem), the most suitable subset of multi-objective optimisation techniques are metaheuristics. Metaheurstics are partial search algorithms designed to provide sufficiently good solutions with limited computational capacity and relative few assumptions made about the problem. Approaches include human mind inspired algorithms (e.g., rough set theory, fuzzy set theory, artificial neural networks [31, 32, 33], evolution inspired algorithms (e.g., genetic algorithms, evolutionary strategies [34, 35]) and swarm intelligence (e.g., ant colony, firefly, particle swarm optimisation [36, 37, 38].

In particular, Genetic Algorithms have been identified as a popular choice for scheduling and allocation related tasks with large, discrete, search spaces [39]. GAs have been proven to be flexible, easy to implement and robust to non-linear and noisy objective functions while requiring minimal information about the search space (e.g., no gradient information required). For example, GAs have been utilised in scheduling for distributed computing systems [40, 41, 42], the deployment of unmanned aerial vehicles [43, 44, 45], human-robot collaboration on tasks [46, 47, 48, 49], optimum resource planning for emergency departments [50, 51] and economics and business [52]. This background demonstrates the applicability of GAs to efficiently allocate a 'batch' of tasks across the currently available workforce, finding a sufficiently good solution for workforce managers. We further discuss the suitability of other metaheuristics (e.g., ant colony, particle swarm optimisation) to find solutions for our multi-objective formal model in Section 5.

3. Methods

In this section, we outline the mathematical formulation of our task allocation model (Section 3.1), including definition of objective functions (Sec-

tion 3.2), along the implementation of the genetic algorithm (Section 3.3), and data used for model validation (Section 3.4).

3.1. Formal Allocation Model

In this section, we frame the problem of allocating tasks across a team of financial workforce of analysts as an optimisation problem. The aim of the optimisation algorithm is to allocate a set of tasks to maximize set objective(s) as defined below. Here, tasks relate to reviewing and resolving identified errors in trading data. These errors can result from human error, technology failures, market volatility and compliance issues. Furthermore, each task can be divided by type with each type having different expectations of completion time, required skills to review and resolve efficiently, and likelihood of being a true error. This is done to address the fact that analysts have a variety of skills and experience which affect their efficiency of completing different types of task. We furthermore assume that tasks are typically allocated in batches across a workforce for the given day. Although the optimisation is only done for a given batch, the model has been adapted to deal with pre-allocation and prior progress on tasks so can be re-run as task lists and operational circumstances change through the day.

In Table 1, we present a summary of notation used in this article. In more detail, let $T = \{1, \ldots, n\}$ be the set of tasks to be allocated and $A = \{1, \ldots, m\}$ the set of (available) analysts in the workforce. Each task $t \in T$ is defined by a number of properties: $\theta_t \in \mathbb{N}^+$ is the task type; $c_t \in \mathbb{R}^+$ the task complexity; $\gamma_t \in [0, 1]$ is the confidence of the task being a true positive (or, more generally, how interesting the task is); and $\pi_t \in \{1, 2, \ldots, p\}$ is the priority (where a lower value means it is higher priority). If $c_t = 1$ this means a task has average (or unknown) complexity, whereas $c_t > 1$ and $c_t < 1$ mean an above-average and below-average complexity respectively.

Similarly, each analyst $a \in A$ has a number of properties: $\eta_{a,\theta} \in \mathbb{R}^+$ is the analyst's efficiency of solving task of type θ; and τ_a is the amount of time (number of seconds) that the analyst is available on the day in question. If $\eta_{a,\theta} = 0$, this means the analyst is unable to execute the task of that type, $\eta_{a,\theta} = 1$ means an average execution time, $\eta_{a,\theta} < 1$ below average, and $\eta_{a,\theta} > 1$ is above average.

A key part of the model is the objective function(s). Several objective functions will be defined below (Section 3.2). For now, we present the objective in general form. Let $T_a \subseteq T$ denote a's assignment, i.e. a set of tasks allocated to analyst $a \in A$. Then, the tuple $\langle T_1, \ldots, T_m \rangle$ denotes an

Symbol	Meaning
n	number of tasks
m	number of analysts
$T = \{1, \ldots, n\}$	set of tasks
$A = \{1, \ldots, m\}$	set of analysts
$T_a \subseteq T$	tasks allocated to analyst a
$\theta_t \in \mathbb{N}^+$	task type
$c_t \in \mathbb{R}^+$	task complexity
$\gamma_t \in [0, 1]$	probability of true positive
p	the number of priorities
$\pi_t \in \{1, 2, \ldots, p\}$	priority of task (lower value is higher priority)
$\eta_{a,\theta} \in \mathbb{R}^+$	analyst a's efficiency of solving task of type θ
τ_a	analyst availability (in number of hours)
$E(t, a)$	expected execution time of task t allocated to analyst a
μ_θ	average task duration of task of type θ
σ_θ^2	execution time variance of task of type θ
$U(T_1, \ldots, T_m)$	Overall utility of an assignment

Table 1: Notation summary

assignment for each analyst. Furthermore, let $U(T_1, \ldots, T_m) \in \mathbb{R}$ denote the overall *utility* or *fitness* of a particular assignment. Given this, the aim is to maximise the assignment utility, which is given by:

$$\langle T_1^*, \ldots, T_m^* \rangle = \arg \max_{T_a \subseteq T, a \in A} U(T_1, \ldots, T_m) \tag{1}$$

subject to each task being allocated to exactly one analyst, i.e., $\forall t \in T : t \in \cup_{a \in A} T_a$ and $\forall a1, a2 \in A, a1 \neq a2 : T_{a1} \cap T_{a2} = \emptyset$

Another crucial aspect of the model is a task's execution time, which depends on both the properties of the task and the analyst. In particular, we assume the *expected* execution time function, $E(t, a)$, is given by:

$$E(t, a) = \frac{\mu_{\theta_t} \cdot c_t}{\eta_{a,\theta_t}} \tag{2}$$

where μ_{θ_t} is the average execution time of a task with type θ_t. Note that, in general, the execution time can also depend on the other tasks allocated. For example, if multiple tasks of the same type are allocated to the same analyst, these can often be completed more efficiently. However, this would make it

challenging to estimate since it adds more unknown parameters. We leave this for future work. Finally, we assume that a task also has an execution time *variance*, denoted by $\sigma^2_{\theta_t}$, indicating the variability in the execution time. For simplicity, we assume this is independent of the allocation.

3.2. Objective Function

A key feature of the framework is that the objective function should be easily changed according to the specific application and client specifications, without having to fundamentally change the model or the algorithm to solve it. In other words, the system should be able to generalise to a wide range of objective functions. In addition, we are able to define *individual* objective functions for every analyst. Depending on specific needs (i.e. different characteristics of individuals leading to increased job satisfaction) these can be tailored to the individual. Here we outline three objective functions appropriate for financial service sector domain, however, this can be easily modified to application-specific needs.

3.2.1. Completion Probability

From the business perspective, a reasonable objective is to maximise the throughput and explicitly optimise efficiency of task completion (e.g., see [12, 13]). An additional complication is that tasks have a sense of priority and it is critical that the highest priority tasks (i.e., those which are particularly time-dependent or most impactful to the NAV) are completed [1, 2]. For those reasons, we aim to maximise the probability of completing tasks, given the analyst's availability, weighted by priority. Maximising completion probability naturally optimises for efficiency (i.e., finding analysts most efficient at a given task), while retaining fairness attributes. For example, framing as a probabilistic utility avoids scenarios where one particularly efficient worker is unfairly overloaded since the probability of them completing a task set tends to zero as more tasks are allocated (in comparison to minimising total completion time). This better aligns the needs and priorities of the business and workers when explicitly considering efficiency [20, 21].

To calculate this, we assume tasks ($T^\pi_a \subseteq T_a$) are executed in order of priority (π) so that tasks in T^π_a are executed before $T^{\pi'}_a$ if $\pi < \pi'$. To calculate the completion probability of a task, we assume execution times are normally and independently distributed with $\phi(x|\mu, \sigma^2)$ denoting the cumulative normal distribution. By defining $\mu^\pi_a = \sum_{t \in T^\pi_a} E(t, a)$ to be the expected total

execution time of tasks, and $(\sigma_t^{\pi})^2 = \sum_{t \in T_a^{\pi}} \sigma_{\theta_t}^2$ as the total variance, the probability of completing all tasks with priority π is then given by:

$$Pr(\pi) = \phi \left(\tau_a \Big| \sum_{\pi'=1}^{\pi} \mu_a^{\pi'}, \sum_{\pi'=1}^{\pi} (\sigma_t^{\pi'})^2 \right) \tag{3}$$

We can then write the conditional probability of completing tasks of priority π given that tasks with $\pi - 1$ (as well as all preceding tasks) are completed as:

$$Pr(\pi | \pi - 1) = \frac{Pr(\pi)}{Pr(\pi - 1)} \tag{4}$$

noting that $Pr(\pi) = Pr(1) \prod_{\pi'=2}^{\pi} Pr(\pi' | \pi' - 1)$. We also note that, given tasks are assumed to be completed in order, its means tasks of priority π are completed only if tasks of priority $\pi - 1$ are already complete. This means that implicitly $Pr(\pi \cap \pi - 1) = Pr(\pi)$ (as denoted in the conditional probability. Now, in order to emphasize high priority tasks compared to lower priority tasks, we define the utility of an *individual* analyst, denoted by $U_a^c(T_a)$ as follows:

$$U_a^c(T_a) = Pr(1) \prod_{\pi'=2}^{\pi} Pr\left(\pi' | \pi' - 1\right)^{1/\pi'} \tag{5}$$

Note that, by adding the power term $1/\pi'$ (conditional) probabilities of lower priority have reduced influence on the utility compared to those with higher priority.

3.2.2. Precision

From the worker perspective, job satisfaction can be (partially) characterised in terms of workload (i.e., avoiding overburden and overtime) along with how engaging or rewarding a given set of tasks are (e.g., [53, 54]). Completion likelihood aims to optimise to reduce potential for overtime but does not consider the fairness of the allocation in terms of rewarding tasks or what workers prefer to do. We now consider parameterisations for both task reward and worker preference. The former is an intrinsic property of the tasks with different types of task likely to be more rewarding, engaging, or fulfilling.

In the context of the financial service sector, the majority of tasks are error checking with different probabilities of identifying a true positive (rather

than a false positive). While error checking tasks resulting in a false-alarm are likely to be perceived as repetitive and unrewarding, identifying true errors has tractable reward for the individual [54]. Each task can be divided by type with each type having different expectations of completion time, required skills to review and resolve efficiently, and likelihood of being a true error, i.e. its *precision* based on historical operations. Therefore, tasks can be scored based on their likelihood of reward (i.e., likelihood of being a true positive γ_t) and considered as an additional objective to optimise. Hence, the aim is to distribute the tasks in a fair manner so that no single analyst has too many false positives.

Given that analysts can be allocated different number of tasks based on their availability, we propose to use the average γ_t of the tasks allocated to analyst to determine their individual utility:

$$U_a^p(T_a) = \frac{1}{|T_a|} \sum_{t \in T_a} \gamma_t \tag{6}$$

This simple objective function can be adjusted to optimise for various task associated scores aiming to increase well-being.

3.2.3. Task Preference

Precision is a property associated with the task, and hence, optimisation can only ensure a fair balance of true-positives are distributed across the workforce. In addition, workers have individual preferences of the types of task they want to complete based on a mixture of personal skills, development goals, and enjoyment of the task itself. Therefore, a more personalised notion of task reward is to directly parameterise which tasks individual analysts *prefer* to complete. In reality, a detailed characterisation of personalised reward (i.e., to explicitly quantify personal skills, development goals and enjoyment) would be a resource intensive procedure and outside the scope of the current work.

As a basic summarisation, we query analysts to respond to a Likert scale for each type of task. These scores are then normalised by worker (i.e., to correct for optimism or pessimism) and we look to explicitly maximise a time-weighted average of these preference scores. This is given by:

$$U_a^t(T_a) = \frac{1}{|T_a|} \sum_{t \in T_a} \zeta_{t,a} \tag{7}$$

where $\zeta_{t,a}$ is the preference score of a given task for the analyst.

3.2.4. Overall Utility

The above measures of individual utility can be combined by considering their product to balance business and worker orientated goals. Here, we define the business orientated utility to be given by equation 5 and the worker orientated utility to be the combination between precision and preference as follows:

$$U_a^w(T_a) = U_a^p(T_a) \cdot U_a^t(T_a) = \frac{1}{|T_a|} \sum_{t \in T_a} \gamma_t \cdot \frac{1}{|T_a|} \sum_{t \in T_a} \zeta_{t,a} \tag{8}$$

Business and worker orientated utility can then be combined into:

$$U_a(T_a) = U_a^c(T_a) \cdot U_a^w(T_a) \tag{9}$$

We note that any combination of completion likelihood, individual preference and precision can be trivially defined. In words, this represents the expected completion time multiplied by the average satisfaction of the tasks (combining precision and preference). Then, to calculate the overall system objective, the individual utilities need to be combined into a single objective. Although the framework allows for different approaches, in this work we assume the overall utility is then computed by the *product* of individual analyst utilities, i.e.:

$$U(T_1, \ldots, T_m) = \prod_{a \in A} U_a(T_a) \tag{10}$$

Using the product is natural here since it indicates the probability of all analysts completing all tasks. In addition, in cooperative game theory literature, this type of joint utility is also known as the Nash product or Nash bargaining solution [55], and has some attractive fairness properties. In particular, it is uniquely characterised by four properties or *axioms*: Pareto efficiency, symmetry, independence of irrelevant alternatives and invariance to affine transformations of the utility function. The latter property means the solution does not rely on interpersonal utility comparisons, i.e. how each individual's utility is scaled. This is particularly important when there is no objective comparison such as money. For more detail, see e.g. [56].

3.3. Genetic Algorithm

To efficiently find solutions to these objective functions we leverage a Genetic Algorithm (GA) implemented by PyGAD [15], an open-source Python

library for GAs. Applied to the domain of task allocation, GAs obtain optimised solutions based on the fitness value (i.e. objective utility functions defined above) from a large pool of candidate solutions (population). A chromosome is a unique solution in the solution space with each chromosome containing n (number of tasks) genes which can take m (number of analysts) discrete values. The best solutions from each population are then passed to the next with variance induced by genetic operators such as mutation (gene alteration) and crossover (gene combination between parent solutions). PyGAD supports a variety of genetic operations and parent selection strategies, making it a flexible solution to allocation problems, and comprehensively identify appropriate hyperparameters (see Section 4.1). In Algorithm 1, we present a pseudocode of our proposed formal model including the GA procedure.

3.4. Simulation

Throughout model development real-world task and workforce data from a global top-10 asset manager was utilised, which this algorithm will service. In Figure 1 we show how our algorithm fits into the workflow of task allocation overseen by a manager. Historical task data is combined with data of the current task list and available analysts, before pre-processing and screening (i.e., identifying if the expected total completion time greatly exceeds availability). Any warnings are sent to the manager who can decide to remove or segment tasks if necessary. This updated task list is then initially allocated by the GA optimiser, which the workforce manager reviews and amends before passing to workforce.

Due to privacy concerns, we publicly evaluate our model using simulated data drawn from typical operating circumstances at the asset manager. To generate estimates of task complexity in Table 2 we define 5 typical types of tasks with varying expected completion times and frequency of occurrence. To simulate test scenarios, we randomly generate a list of tasks with occurrence frequency and expected completion times drawn from this table. We assume task duration and variance by type to be normally distributed.

To calculate the expected execution time function given in equation 2 and emulate the differences in skills of analysts we modify $\eta_{a,\theta}$ (efficiency by task type) randomly by a factor between 0.9 - 1.1 when generating a simulated analyst. We note that setting uniform analyst efficiencies does not impact the ability of the model to converge. In practice analysts will also balance newly allocated tasks with left-over work from the prior workday. To emulate

Algorithm 1 Pseudocode of the Formal Model with GA generated solutions.

START: PREPROCESS(τ, μ)
// **Evaluate Problem**
Compute total sum of workforce availability τ_{total};
Compute total sum of expected task duration μ_{total};
Identify tasks with prior allocation and completion and adjust τ_{total} and μ_{total} appropriately;

// **Select Task List**
If $\tau_{total} >> \mu_{total}$ drop lowest priority tasks;

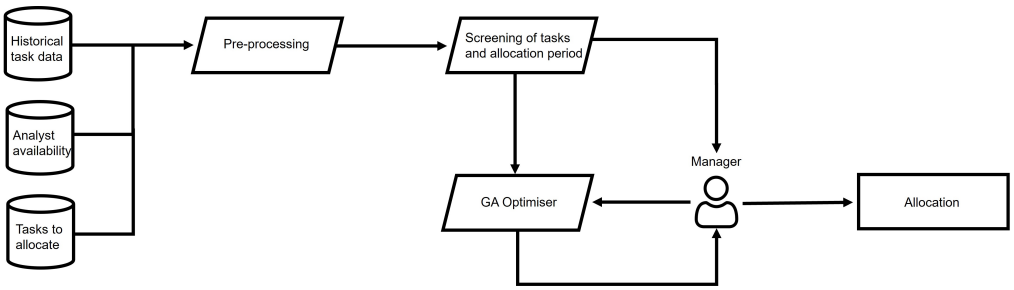
START: Find solution with GA(j, χ, η)
// **Initialise Generation**
$k := 0$;
$P_k :=$ population of j randomly generated solutions
// **Evaluate** P_k
Compute fitness(i) for each $i \in P_k$;
// **Iterate to find best possible solution in 50 generations**
while $k < 50$ **do**
    // **Create generation k + 1**
    // **1. Elitism**
    Select $(1–\chi) \times j$ members of P_k and insert into P_{k+1};

    // **2. Crossover**
    Select $\chi \times j$ members of P_k (steady state);
    pair them up and produce offspring;
    insert the offspring into P_{k+1};

    // **3. Mutate**
    Random mutation applied at gene level;
    High adaptive mutation rate for $\eta \times j$ members with lowest fitness
    Low adaptive mutation rate for $(1-\eta) \times j$ members with highest fitness

    // **Evaluate** P_{k+1}:
    Compute fitness(i) for each $i \in P_k$;

    // **Increment:**
    k := k + 1
end while



Source: Authors.

Figure 1: Flow diagram showing the working context of our proposed model. Historical data on previous tasks and analysts is first combined with data on the current allocation period. This is pre-processed to estimate parameters for the current allocation period (e.g., expected completion type of tasks to be completed). The task list and analyst availability is screened to identify significant overall burden (i.e., expected total completion time greatly exceeds availability) or tasks which are individually too long to complete in the period. Any warnings are sent to the manager who can decide to remove or segment tasks if necessary. This updated task list is then initially allocated by the GA optimiser, which the workforce manager reviews and amends before passing to workforce.

this, we pre-allocate 5% of our simulated tasks with a randomly generated prior progress time.

To reflect the heavy workload found in asset management, we define test scenarios that are 'difficult but achievable'. Practically this means defining a task list which is expected to between 1.01 and 1.1 times the total analyst availability. Due to the nature of a probabilistic completion time fitness function, complete over-burden can result in zero probability for an analyst and the algorithm failing to converge. For this reason, problems (or task lists) should be pre-screened to ensure they meet this criterion. For problems with low workload, we heavily penalise allocations with analysts that have zero allocated tasks to ensure a fair distribution is still found.

4. Results

This section demonstrates the performance of our allocation model with several GA hyperparameter set-ups relative to baseline allocation algorithms. We also consider the run-time complexity of our approach and further evaluate our formal model on a set of realistic use cases of different size and optimising goals (i.e. fitness function). For comparison between hyperparameter choices and to baseline algorithms, we simulate a basic test scenario of 65

Type, θ_t	μ_θ (seconds)	σ^2_θ (seconds)	Relative frequency
A_t	1800	90000	1
B_t	3600	810000	0.75
C_t	7200	1440000	0.5
D_t	14400	7290000	0.25
E_t	21600	12960000	0.05

Table 2: Properties of five tasks types which are used as a basis to simulate real-world scenarios. Mean duration (μ_θ) and variance (σ^2_θ) are the expected global completion (and variance) time for a task of that type across all analysts. Relative frequency defines the proportion of each task type represented in a simulated scenario.

tasks to be allocated between 10 analysts to optimise for priority weighted completion time. To understand how our model scales, we adopt Big $\mathcal{O}$ notation to describe the limiting (worst case) behaviour. We further simulate test scenarios up-to a size of 325 tasks and 50 analysts for both single (completion probability only) and multi (both completion probability and true positive probability) objective functions to evaluate empirically.

4.1. Hyperparameter Selection

To select an effective combination of GA hyperparameters we validate performance for a variety of population sizes along with different parent selection, mutation, and cross-over techniques. In Figure 1 (left panel) we show the average performance (as defined by total utility) of three distinct hyperparameter approaches; green: steady state parent selection with adaptive mutation, blue: tournament parent selection with adaptive mutation; red: steady state parent selection with scramble mutation. All other hyperparameters are consistent as defined in Table 3. Our key findings are as follows:

- **Population size and number of generations**: 50 generations with 500 solutions per population reliably result in a converged solution (as shown by Figure 1 right panel). Increasing the number of generations (at the cost of population size) generally results in decreased performance, however effect is minimal.

- **Mutation**: Adaptive mutation results in the best performance. As outlined in [57], the weak point of 'classical' GAs is that mutation is randomly applied to all chromosomes, irrespective of their fitness.

Adaptive mutation solves this by applying a high (low) mutation rate to low (high) fitness solutions overcoming the usual trade-off between making incremental improvements and finding better maxima. Scramble mutation (blue) performs particularly poorly since it keeps the number of tasks assigned per analyst constant (similar for inversion and swap mutation).

- **Parent Selection**: Steady State (green) outperforms Tournament parent selection (blue). Along with defined elitism, there may be minimal selection of low fitness solutions to be passed to the next generation, highlighting the importance of being able to define a high mutation rate and escape local maxima.

- **Crossover**: Crossover choice had minimal effect on performance for this use case, leading to single point crossover being selected.

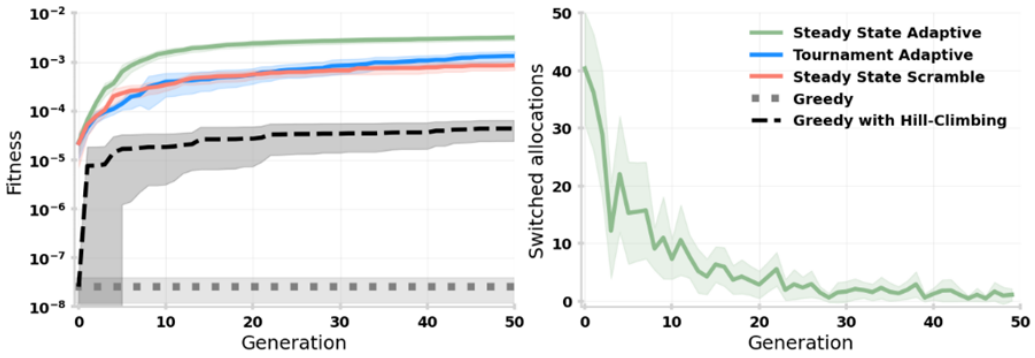


Figure 2: (Left) Average performance (as defined by fitness of best solution) of genetic algorithms (solid lines; green, blue, red) and baseline algorithms (greedy – dotted grey, greedy and hill-climbing – dashed black). The average is found by performing 20 runs of each algorithm set-up and the shaded regions represent the 95% confidence interval. (Right) Average number of tasks switching allocation per generation for best solution. Only shown for best GA hyperparameter set-up (green - Table 3).

Our final hyperparameters are described in Table 3 (represented by the green line in Figure 2). For larger scale problems, we utilise the same hyperparameters however increasing the population size, number of parents mating and elitism proportionally to the number of tasks.

Population Size	500
Generations	50
Crossover Type	Single Point
Mutation Type	Adaptive (Random with variable rates)
Mutation Probability	[0.9, 0.05]
Parents Mating	50
Elitism	10

Table 3: Hyperparameters for fiducial GA run validated on basic test scenario of 65 tasks allocated across 10 analysts.

4.2. Comparison to baseline heuristics

To validate our choice of a metaheuristic such as a Genetic Algorithm for efficiently finding allocation solutions to our formal model, in Figure 2 (left panel) we directly compare to baseline heuristics.

We adopt a greedy allocation policy [58] (grey dotted line), which allocates sequentially by task to the analyst which maximises global utility at each step. Generally, a greedy policy in optimisation problems does not produce an optimal solution, however can yield local maxima with reasonable run-times. Despite being computationally efficient, this typically provides a poor scoring solution and often fails when applied to more complex fitness functions. We improve upon the initial greedy allocations through application of a hill climbing policy (black dashed line). Hill climbing is an iterative search optimisation technique which starts with an arbitrary solution, then makes incremental changes to find better local solutions. Here, the hill climbing policy randomly swaps two task allocations to find an improved solution (from the initial greedy solution). The GA optimisation significantly outperforms both the baseline greedy algorithm and the hill climbing improvement (scaled to be same number of utility function evaluations as the GA).

4.3. Stopping condition

To validate an appropriate stopping condition (or number of generations) for our GA, we consider both the convergence of fitness score (Figure 2 left panel) and the number of task allocations changed between best solutions of each generation (right panel). We note that fitness (and changes to best allocation) changes rapidly in the first 10 generations, settling towards generation 20 and showing steady but minimal improvement (and changes to

best allocation) towards generation 50. We find 50 generations is an appropriate stopping point to ensure a high fitness solution is consistently found, however, if a quicker compute time was required, solutions from generation 20 onwards could be used.

4.4. Scalability

We now consider the run-time complexity of our formal model supported by a GA optimiser. We adopt Big $\mathcal{O}$ notation to describe the limiting (worst case) behaviour of our approach as $n \to \infty$ (where n is the size of the population in the GA). The key operations and time-complexity are as follows: i) initialisation of population: $\mathcal{O}(n)$, ii) evaluation of utility: $\mathcal{O}(n^2)$ (since utility function evaluation scales with $\mathcal{O}(n)$ and must be evaluated for each worker whose number we assume scales with the number of tasks and hence population size n), iii) selection $\mathcal{O}(nlogn)$ (requires sorting), iv) crossover $\mathcal{O}(n)$, v) mutation $\mathcal{O}(n)$ and vi) termination conditions $\mathcal{O}(1)$ (since is manually defined to be 50 generations). The dominating term is therefore $\mathcal{O}(n^2)$.

In Figure 3, we also empirically estimate the run-time scaling for our allocation model for a set of realistic scenarios with increasing numbers of tasks and analysts for single (black) and multi (i.e, completion likelihood and true positive likelihood; grey) objective problems. The absolute run-time units are seconds (based on serial evaluation on a single node cluster with 32GB of memory). In response to the increasing problem scale, we increase the population size of our GA search space proportionally to the number of tasks to allocate (i.e. double number of tasks equates to double the population size). Despite this linear scaling, the number of fitness function evaluations also increase due to a greater pool of analysts. We fix the number of generations to be 50 for each problem size, finding solutions to be well converged. For a given problem size, increasing the number of generations would lead to a linear increase in computation time. This is in-keeping with our theoretical complexity scaling limit of n^2.

We note that this holds one key assumption that solutions converge consistently when increasing the scale of the GA search space linearly with the number of tasks to be allocated. In reality for larger use cases, it would be more suitable to scale the population size proportional to both the increase in workers and tasks, therefore our formal model would instead scale $\mathcal{O}(n_{task}^3)$ (where here n_{task} is the number of tasks to allocate).

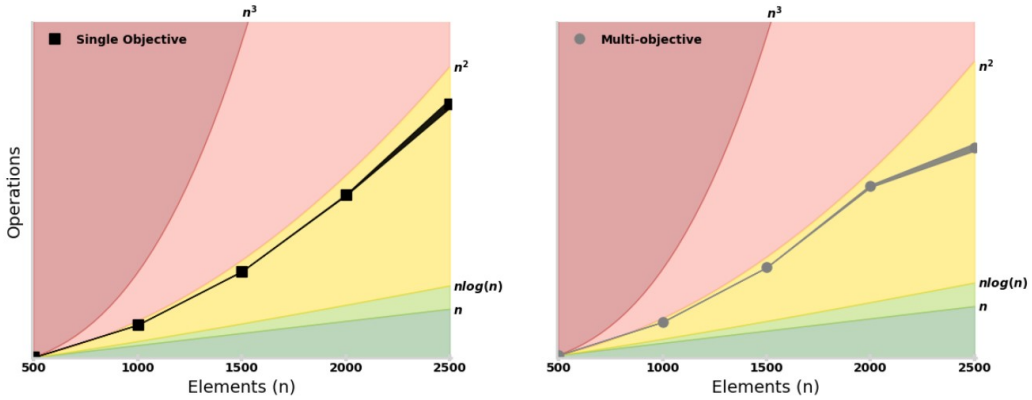


Figure 3: Run-time complexity for single objective (left) and multi-objective (right) allocations for a range of problem sizes, as denoted by the GA population size (n). Connecting lines show the width of the 95% CI. The background is shaded by scaling complexity (relative to smallest scale problem in each panel; i.e. 65 tasks and 10 analysts) divided by n, $nlog(n)$, n^2, n^3 (in order of increasingly poor scaling).

4.5. Comparison to current practice

We now evaluate our formal model relative to typical operating circumstances (i.e., task allocation performed by workforce manager) to assess its real-world benefit. To emulate how a manager may allocate tasks across a typical workforce we consider two manager strategies:

1. **Efficiency**. Here the manager allocates tasks one-by-one (in order of priority) to the analyst who has historically performed that type of task most efficiently. Once a given analyst is over-burdened, following tasks are then allocated to the second most efficient analyst, and so on. This is effectively the greedy algorithm approach described above. This allocation strategy also is designed to minimise potential impact to the NAV by allocating the highest priority (and hence highest materiality) tasks one-by-one.
2. **Balancing task numbers**. Here the manager allocates (pseudo-randomly) to ensure each analyst has roughly the same number of tasks, a basic notion of fairness. This approach ignores other measures of reward, the overall length of tasks, and the likelihood of each analyst completing their allocation.

In Figure 4, we compare the simulated manager allocations against our single and multi-objective GA approaches. Scores for the GA approaches are found

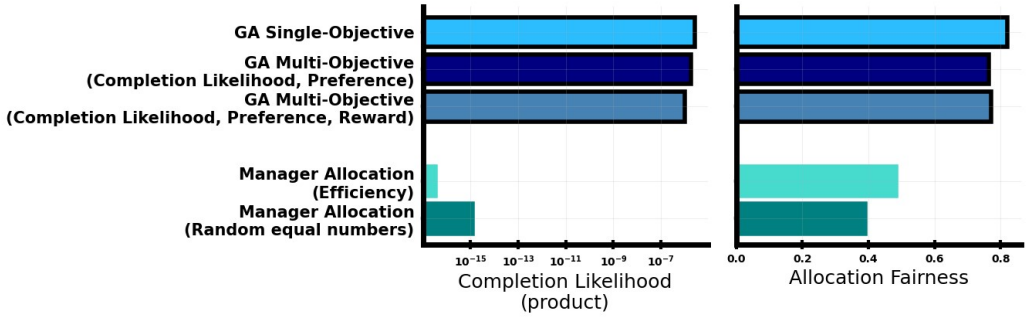


Figure 4: Comparison between simulated manager-led allocations and the proposed formal model. In each panel a simulated manager allocating to maximise efficiency (light green), and, to balance task numbers (dark green) is compared to a single-objective GA to optimise for completion likelihood (light blue), a multi-objective GA optimising for completion likelihood and task preference (dark blue) and a multi-objective GA optimising for completion likelihood, task preference and reward (grey blue). The scores for GAs solutions are found from the average of 5 runs. (Left) Completion likelihood across complete workforce (i.e., product of individual completion likelihoods for their allocation). (Right) Allocation fairness. This is maximum difference between the best and the worst allocation by analyst. A given analyst's allocation is evaluated by the product of completion likelihood, the time-weighted preference score and the time-weighted reward (anomaly score).

from the average of 5 runs. Here we use a test scenario with 65 tasks allocated across a workforce of 10 analysts (as described in Section 4.1). For each task we generate a 'reward' score based on the anomaly score of the task. For each analyst we also generate 'preference' scores by task-type. Each GA optimises completion likelihood along with preference (navy) and both preference and reward (grey blue).

In the left panel we quantify the success of simulated managers and GAs at maximising the likelihood of completing the complete set of tasks. This is effectively the business orientated goal. This is found by the product of completion likelihood across all 10 analysts (i.e., equation 10). As discussed in Section 3.2, we use the product rather than the average or sum to ensure fairness in evaluation. We find all GA approaches offer significant improvement over the simulated manager allocations, increasing global likelihood of tasks being completed by at-least 7 orders of magnitude. In particular we note the poor performance of the simulated manager allocating tasks to the most efficient analyst; highlighting the pitfalls of greedy policies (even relative to a pseudo-random allocation).

In the right panel we quantify the overall fairness of the allocation across

the workforce. For each analyst the 'quality' of their allocation is found by the product of the completion likelihood, the (time-weighted) average preference score, and the (time-weighted) average anomaly score (reward). This score balances both worker over-load and reward of tasks. The fairness of each allocation is then found by finding the difference between the best and the worst scoring allocation by-analyst. We find that both single and multi-objective GA approaches result in significantly fairer allocations relative to the simulated manager allocations.

5. Discussion

In practice, AI powered allocation and scheduling recommendations will be used with resource managers in-the-loop (i.e., human-in-the-loop; HITL). HITL integrates human judgement and expertise into the decision making process, augmenting the capabilities of decision-support tools, such as our proposed formal model. Human involvement in the decision-making process fosters trust and transparency, as ultimately the resource manager has final say. Another critical factor, is the contextual domain knowledge of resource managers, often difficult to capture in modelling. Practically this AI recommendation would therefore be used as a starting point for the manager to 'tweak' depending on current operational circumstances and personal domain knowledge (e.g., escalating certain tasks to be more critical than identified in the system).

In Figure 5, we show a visualisation containing an example allocation found from our GA model applied to the multi-objective test scenario with 65 tasks and 10 analysts. In this instance, a manager may identify high priority tasks which are scheduled to be completed later in the day (e.g., Analyst 3's 4th task) and choose to re-allocate to ensure they are completed in a timely manner (e.g., swap with Analyst 10's first task). While HITL likely reduces the utility score of an allocation, it still offers the best approach to maximise business and well-being (significantly outscoring current working practice), while ensuring the hybrid approach is robust by avoiding pitfalls not captured by modelling. Additionally, human input may enable soft and hard constraints on allocation to be identified (e.g., that a given analyst cannot a specific task due to external factors not previously captured in data). Constrained optimisation (and hence narrowing of the search space) potential enables solutions to be found faster.

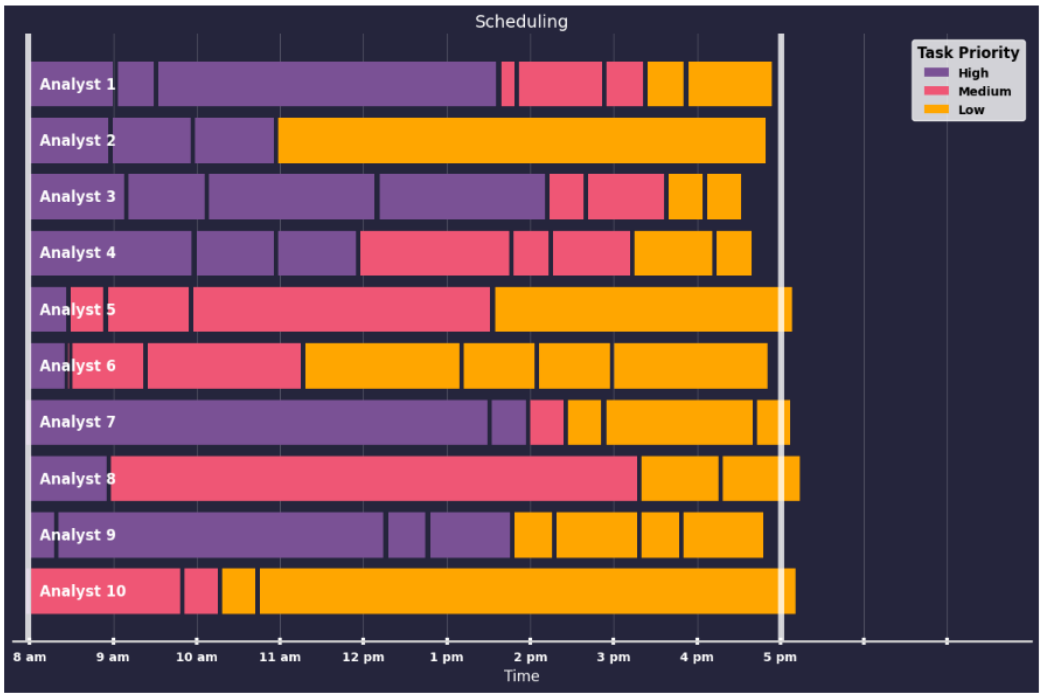


Figure 5: Example task allocation and scheduling found from the multi-objective test case with 65 tasks and 10 analysts. Each row shows the allocation for a given analyst throughout their given workday. Tasks are coloured by priority (purple: high, pink: medium, orange: low). The available workhours are given by the white solid lines. Note: work breaks are not explicitly drawn onto the schedule with expected completion times compensating for breaks through the day.

Another important purpose of the decision-support system is to identify significant pressures on the workforce (i.e., when even optimised solutions may be expected to take significantly longer than the total available workhours for a team). Providing insights into the available allocations, along with specifics about the severity of the potential overload (or underload) is therefore critical to inform business processes (e.g. whether the situation can be solved by over-time or severe enough to consider bringing in analysts from external teams).

An important adaptation to our formal model, is the ability to consider tasks which are pre-allocated or have prior completion. As a result, we are able to periodically (or on-demand) re-evaluate the allocation and provide updated recommendations. Throughout the day progress will naturally fluc-

tuate by analyst relative to initial expectations and operational circumstances may change. Therefore, the ability to efficiently re-optimise task allocations to improve business and well-being needs is critical. Additionally if significant pressures are identified and external members are brought in to help, again, quick re-allocation is required.

In this manuscript, we have demonstrated one example of a metaheuristic (i.e., GA) can quickly converge on solutions with significant improvements on current working practice. Combinatorial optimisation with multiple objectives makes GA a popular choice, however, we note the validity of other approaches such as swarm intelligence (e.g., firefly, particle swarm optimisation: PSO) which have shown increasing popularity in recent years. In the instance of PSO, we note one potential advantage is the ability for faster convergence and more diversity in search trajectories (due to momentum effects) [59]. A promising variant of GAs for faster convergence have been those with adaptive mutation and crossover rates (e.g., [60]). Here, we demonstrate taking an adaptive mutation strategy (dependent on the fitness of the solution) results in faster convergence on higher quality solutions relative to other strategies (e.g., scramble). We look to future work to compare the performance of other metaheuristics utilising our formal model.

Finally, we demonstrate that a formal model explicitly optimising for well-being can still find solutions which still significantly improve efficiency relative to current working practice (Figure 4). Even with potentially conflicting goals, this approach recognises the two-way psychological relationship between employer and employee, outside of the legal and formal framework that an employee simply obtains work for fiscal reward [22]. Developing working practices that better the 'pyschological contract' between employee and employer (e.g., allowing workers to define preferred tasks) enables workers to view their relationship as a two-way transaction, rather than one imposed against their interests [20]. In turn, finding solutions which improve efficiency benefit both the business and workers by reducing the potential overburden of overtime. In future work we look to develop more detailed mathematical representations of well-being which quantify factors such as short-term satisfaction and personal skill development.

6. Conclusions

Financial service sector companies routinely check and resolve errors across huge data sets, putting significant pressure on workforces with in-

creasingly manual tasks. Automation is critical to ensure tasks are allocated efficiently and fairly. In this article we introduced a formal model leveraging a GA to allocate and schedule tasks across a workforce. Through definition of objective functions for completion probability and worker reward, we present a methodology to allocate tasks to both maximise business orientated goals and worker well-being. The GA significantly outperforms baseline heuristics and scales intermediately to a range of typical working circumstances, demonstrating the potential for metaheuristic approaches. This formal model is intended to service workforces at a global top 10 asset manager. Finally, we note the applicability of our formal model to other domains, in particular medical triage within an emergency department which we look to in future work.

References

[1] B. Suh, A systematic approach to predicting nav errors, Retrieved from (2024).
URL https://www.oncorps.ai/briefs/nav-errors

[2] F. Europe, Preventing the "huge consequences" of nav oversight error, Retrieved from (2021).
URL https://funds-europe.com/the-huge-consequences-of-nav-oversight-error/

[3] U. Labor, Financial analysts - work environment, us bureau of labor statistics, Retrieved from (2023).
URL https://www.bls.gov/ooh/business-and-financial/financial-analysts.htm#tab

[4] G. Sachs, Working conditions survey, Retrieved from (2022).
URL https://assets.bwbx.io/documents/users/iqjWHBFdfxIU/rSEzNw9cve9E/v0

[5] WSO, Wso 2021 investment banking work-conditions survey, Retrieved from (2021).
URL https://www.wallstreetoasis.com/forum/investment-banking/wso-2021-investm

[6] U. Labor, Job openings and labor turnover summary, us bureau of labor statistics, Retrieved from (2022).
URL https://www.bls.gov/news.release/jolts.nr0.htm

[7] A. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, Staff scheduling and rostering: A review of applications, methods and models, European

Journal of Operational Research 153 (1) (2004) 3–27, timetabling and Rostering. doi:https://doi.org/10.1016/S0377-2217(03)00095-X.
URL https://www.sciencedirect.com/science/article/pii/S037722170300095X

[8] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, L. De Boeck, Personnel scheduling: A literature review, European journal of operational research 226 (3) (2013) 367–385.

[9] A. Ahmadian Fard Fini, A. Akbarnezhad, T. H. Rashidi, S. T. Waller, Job assignment based on brain demands and human resource strategies, Journal of Construction Engineering and Management 143 (5) (2017) 04016123.

[10] A. Starkey, H. Hagras, S. Shakya, G. Owusu, A genetic algorithm based approach for the simultaneous optimisation of workforce skill sets and team allocation, in: Research and Development in Intelligent Systems XXXIII: Incorporating Applications and Innovations in Intelligent Systems XXIV 33, Springer, 2016, pp. 253–266.

[11] D. A. Nembhard, F. Bentefouet, Selection policies for a multifunctional workforce, International journal of production research 52 (16) (2014) 4785–4802.

[12] D. Mourtzis, V. Siatras, J. Angelopoulos, N. Panopoulos, An intelligent model for workforce allocation taking into consideration the operator skills, Procedia CIRP 97 (2021) 196–201.

[13] V. Derkinderen, J. Bekker, P. Smet, Optimizing workforce allocation under uncertain activity duration, Computers & Industrial Engineering 179 (2023) 109228.

[14] F. B. Review, Companies need to know the dollar cost of employee turnover, Retrieved from (2018).
URL https://www.forbes.com/sites/billconerly/2018/08/12/companies-need-to-kno

[15] A. F. Gad, Pygad: An intuitive genetic algorithm python library (2021). arXiv:2106.06158.

[16] J. E. Gomar, C. T. Haas, D. P. Morton, Assignment and allocation optimization of partially multiskilled workforce, Journal of construction Engineering and Management 128 (2) (2002) 103–109.

[17] B. Shahbazi, A. Akbarnezhad, D. Rey, A. Ahmadian Fard Fini, M. Loosemore, Optimization of job allocation in construction organizations to maximize workers' career development opportunities, Journal of Construction Engineering and Management 145 (6) (2019) 04019036.

[18] P. Cowling, N. Colledge, K. Dahal, S. Remde, The trade off between diversity and quality for multi-objective workforce scheduling, in: Evolutionary Computation in Combinatorial Optimization: 6th European Conference, EvoCOP 2006, Budapest, Hungary, April 10-12, 2006. Proceedings 6, Springer, 2006, pp. 13–24.

[19] G. Quan, G. W. Greenwood, D. Liu, S. Hu, Searching for multiobjective preventive maintenance schedules: Combining preferences with evolutionary algorithms, European Journal of Operational Research 177 (3) (2007) 1969–1984.

[20] E. Demerouti, A. B. Bakker, F. Nachreiner, W. B. Schaufeli, The job demands-resources model of burnout., Journal of Applied psychology 86 (3) (2001) 499.

[21] H. Bao, C. Liu, J. Ma, J. Feng, H. He, When job resources function as a stress buffer: A resource allocation perspective of the job demands-resources model, Personality and Individual Differences 192 (2022) 111591.

[22] A. B. Bakker, E. Demerouti, Job demands–resources theory: Taking stock and looking forward., Journal of occupational health psychology 22 (3) (2017) 273.

[23] T. L. Saaty, What is the analytic hierarchy process?, Springer, 1988.

[24] P. Vincke, Multicriteria decision-aid, John Wiley & Sons, 1992.

[25] G.-H. Tzeng, J.-J. Huang, Multiple attribute decision making: methods and applications, CRC press, 2011.

[26] J.-P. Brans, P. Vincke, Note—a preference ranking organisation method: (the promethee method for multiple criteria decision-making), Management science 31 (6) (1985) 647–656.

[27] R. L. Keeney, H. Raiffa, Decisions with multiple objectives: preferences and value trade-offs, Cambridge university press, 1993.

[28] J. R. Eastman, H. Jiang, J. Toledano, Multi-criteria and multi-objective decision making for land allocation using gis, Multicriteria analysis for land-use management (1998) 227–251.

[29] S. L. Gebre, D. Cattrysse, J. Van Orshoven, Multi-criteria decision-making methods to address water allocation problems: A systematic review, Water 13 (2) (2021) 125.

[30] C. Shyalika, T. Silva, A. Karunananda, Reinforcement learning in dynamic task scheduling: A review, SN Computer Science 1 (6) (2020) 306.

[31] Z. Pawlak, Rough sets, International journal of computer & information sciences 11 (1982) 341–356.

[32] H.-J. Zimmermann, Fuzzy set theory—and its applications, Springer Science & Business Media, 2011.

[33] S. J. Russell, P. Norvig, Artificial intelligence a modern approach, London, 2010.

[34] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.

[35] H.-G. Beyer, H.-P. Schwefel, Evolution strategies–a comprehensive introduction, Natural computing 1 (2002) 3–52.

[36] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE transactions on systems, man, and cybernetics, part b (cybernetics) 26 (1) (1996) 29–41.

[37] X.-S. Yang, Nature-inspired metaheuristic algorithms, Luniver press, 2010.

[38] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, Vol. 4, IEEE, 1995, pp. 1942–1948.

[39] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, et al., A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, Journal of Parallel and Distributed computing 61 (6) (2001) 810–837.

[40] D. P. Vidyarthi, A. K. Tripathi, Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm, Journal of Systems Architecture 47 (6) (2001) 549–554.

[41] P. Rekha, M. Dakshayini, Efficient task allocation approach using genetic algorithm for cloud environment, Cluster Computing 22 (4) (2019) 1241–1251.

[42] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, M. U. Chowdhury, An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments, Neural Computing and Applications 32 (2020) 1531–1541.

[43] G. Chen, J. B. Cruz, Genetic algorithm for task allocation in uav cooperative control, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2003, p. 5582.

[44] W. Zhu, L. Li, L. Teng, W. Yonglu, Multi-uav reconnaissance task allocation for heterogeneous targets using an opposition-based genetic algorithm with double-chromosome encoding, Chinese Journal of Aeronautics 31 (2) (2018) 339–350.

[45] X. Wu, Y. Yin, L. Xu, X. Wu, F. Meng, R. Zhen, Multi-uav task allocation based on improved genetic algorithm, IEEE Access 9 (2021) 100369–100379.

[46] F. Chen, K. Sekiyama, T. Fukuda, A genetic algorithm for subtask allocation within human and robot coordinated assembly, in: 2012 International Symposium on Micro-NanoMechatronics and Human Science (MHS), IEEE, 2012, pp. 507–511.

[47] E. Nunes, M. Gini, Multi-robot auctions for allocation of tasks with temporal constraints, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 29, 2015, p. 1.

[48] A. Raatz, S. Blankemeyer, T. Recker, D. Pischke, P. Nyhuis, Task scheduling method for hrc workplaces based on capabilities and execution time assumptions for robots, CIRP Annals 69 (1) (2020) 13–16.

[49] Y. Y. Liau, K. Ryu, Genetic algorithm-based task allocation in multiple modes of human–robot collaboration systems with two cobots, The International Journal of Advanced Manufacturing Technology 119 (11-12) (2022) 7291–7309.

[50] J.-Y. Yeh, W.-S. Lin, Using simulation technique and genetic algorithm to improve the quality care of a hospital emergency department, Expert Systems with Applications 32 (4) (2007) 1073–1083.

[51] M. Yousefi, M. Yousefi, R. P. M. Ferreira, J. H. Kim, F. S. Fogliatto, Chaotic genetic algorithm and adaboost ensemble metamodeling approach for optimum resource planning in emergency departments, Artificial intelligence in medicine 84 (2018) 23–33.

[52] M. Sieja, K. Wach, The use of evolutionary algorithms for optimization in the modern entrepreneurial economy: interdisciplinary perspective, Entrepreneurial Business and Economics Review 7 (4) (2019) 117–130.

[53] S. A. DeVaney, Z. Chen, Job satisfaction of recent graduates in financial services, Retrieved on 12 (12) (2003) 2006.

[54] B. Sypniewska, Evaluation of factors influencing job satisfaction, Contemporary economics 8 (1) (2014) 57–72.

[55] J. F. Nash Jr, The bargaining problem, Econometrica: Journal of the econometric society (1950) 155–162.

[56] K. Binmore, Bargaining and fairness, Proceedings of the National Academy of Sciences 111 (supplement_3) (2014) 10785–10788.

[57] S. Marsili Libelli, P. Alba, Adaptive mutation in genetic algorithms, Soft computing 4 (2000) 76–80.

[58] E. W. Dijkstra, A note on two problems in connexion with graphs, in: Edsger Wybe Dijkstra: His Life, Work, and Legacy, 2022, pp. 287–290.

[59] L. Goel, An extensive review of computational intelligence-based optimization algorithms: trends and applications, Soft Computing 24 (2020) 16519–16549.

[60] S. Han, L. Xiao, An improved adaptive genetic algorithm, in: SHS Web of Conferences, Vol. 140, EDP Sciences, 2022, p. 01044.