Multiobjective evolutionary algorithms for strategic deployment of resources in operational units

John H. Drake[a,*], Andrew Starkey[b], Gilbert Owusu[b], Edmund K. Burke[c]

[a] *Operational Research Group, Queen Mary University of London, Mile End Road, London E1 4NS, UK*
[b] *Business Modelling and Operational Transformation Practice, British Telecom, Adastral Park, Martlesham Heath, Ipswich, UK*
[c] *University of Leicester, University Road, Leicester, LE1 7RH, UK*

Abstract

Large-scale infrastructure networks require frequent maintenance, often performed by a team of skilled engineers spread over a large area. The set of tasks allocated to an engineer can have a huge impact on overall efficiency, whether that be in terms of time taken to complete all tasks, staffing costs or environmental costs in terms of emissions. When required to efficiently allocate a set of geographically distributed tasks to a maintenance engineering workforce, one approach is to define working areas for which teams of engineers are responsible. Often a key obstacle to overcome when looking for solutions is ensuring a balance between multiple competing objectives. In this paper, we employ a number of multiobjective evolutionary algorithms to analyse a simulation model for a real-world workforce optimisation problem used by BT. We provide a detailed analysis of the class of problems to be solved, where the workforce and a set of service distribution points must be split into smaller working areas, referred to as operational units. As the choice of how many operational units to split a larger working area into is critical, some of the practical considerations that must be made when addressing such problems are highlighted. This research has allowed the planning team at BT to understand the unique complexities of the nature of the problems they face in different areas of the UK, particularly with respect to the choice of number of operational units, and has strengthened their ability to design operational units effectively.

Keywords: Decision support systems, Workforce Optimisation, Multiobjective optimisation, Multiobjective Evolutionary Algorithms, Work Area Optimisation

1. Introduction

In large organisations that have a large base of employees for service distribution, effective task allocation is critical to ensuring cost-efficient service delivery. Poor allocation of tasks can lead to increased costs, reduced revenue and increased dissatisfaction for both customers and employees. For companies that manage large infrastructure networks, such as those found in rail, telecommunications or power distribution, it is often desirable to divide large geographical areas

*Corresponding author
Email addresses: j.drake@qmul.ac.uk (John H. Drake), andrew.starkey@bt.com (Andrew Starkey), gilbert.owusu@bt.com (Gilbert Owusu), edmund.burke@leicester.ac.uk (Edmund K. Burke)

into more manageable operational units when allocating tasks. In the case of BT, the industrial partner for this project, not only are smaller operational units preferable from a management perspective, they can foster a greater level of problem ownership from employees. This is due to the increased level of familiarity with their own unit, and competition with teams responsible for other units.

Typically such problems create a computational search space that makes exhaustive enumeration impractical within a reasonable amount of time, leading to widespread use of metaheuristic techniques. Pour et al. [18] presented a hyper-heuristic framework for the allocation of maintenance tasks in the Danish rail network. Their work used multiple heuristics to define the operational unit that each individual maintenance engineer was responsible for. Peng and Ouyang [17] took a different approach, first clustering maintenance tasks into smaller projects, before assigning projects to specific teams. Also in the context of railway maintenance, Gorman and Kanet [8] developed a hybrid Constraint Programming and Genetic Algorithm approach, scheduling maintenance tasks for a 'gang' of engineers moving around the network. Heuristics and metaheuristics have also been used to solve the related traveling repairman problem [5, 19]. However, these methods focused on finding the best route for a single engineer completing a set of known tasks.

We consider the problem of defining the working areas, or operational units, for teams of maintenance engineers working on a telecommunications network in the UK. Previous work introduced a simulation model to calculate an estimate of the performance of different operational unit designs using a type-2 fuzzy logic [10] based system [21]. This model underpins the strategic organisational design tool used at BT to design operational units. As this tool is responsible for organising the workload of over 15,000 engineers in the UK, determining the best search strategy for the planning team is key. Previously, redesigns were only carried out every few years. However, the introduction of this tool has reduced the length of time required to design operational units from a few months to a few hours, freeing up managers to tackle other field issues. As a result of the now very short redesign process, a new quarterly review strategy has been implemented. Additionally, as the volume of demand for a particular service distribution point and the volume of supply of service in terms of the availability of different resources can vary over time, frequent review of operational units can lead to immediate performance and efficiency gains. In addition to performing the process regularly, the planning team may wish to consider multiple variations of the same problem by changing certain external parameters, reflecting different managerial decisions. This makes metaheuristic search techniques attractive when tackling this problem, as such approaches aim to provide high quality solutions in a limited amount of computational time. We provide more extensive results and analysis than previous work, particularly focusing on the effects different managerial decisions can have on the complexity of the problem and the interaction between primary objectives. One of these decisions is the initial choice of the number of operational units each area is to be split into, which has a great impact on both the search space of problem and the computational time required to evaluate the model.

In practice, many real-world problems have more than one objective. Often these objectives are competing, with an improvement in one objective only possible to the detriment of another. Some of the most widely used and successful methods to solve such problems are multiobjective evolutionary algorithms [3]. Based on the natural notion of evolution, a population of solutions to the problem are 'evolved' through means of natural selection and variation. Here we apply a set of multiobjective evolutionary algorithms to a real-world optimisation problem, based on a previously developed interval type-2 fuzzy logic simulation [21]. Previous work has used multiobjective

evolutionary algorithms to solve the problem of operational unit design [21, 20]. However, their experimentation was limited to a single method, using few fitness evaluations. We provide a more comprehensive set of experiments, using a wider variety of multiobjective evolutionary algorithms. This investigation will provide a clear understanding of the impact of using different types of multiobjective evolutionary algorithms to address this problem, allowing strategic decisions to be made about which to use in the current real-world system.

The remainder of the paper is organised as follows: Section 2 describes the real-world optimisation problem studied in this paper and the simulation model used to assess the quality of solutions. Section 3 discusses multiobjective optimisation problems and introduces the multiobjective evolutionary algorithms used in our experimentation. In Section 4 we analyse different aspects of the existing simulation model, and the effect that it has on the search process. Section 5 presents the results of applying different multiobjective evolutionary algorithms to problem instances with different characteristics, using different numbers of operational units. Finally, Section 6 provides some concluding remarks and potential directions for further research.

2. Real-world planning of operational units

Workforce optimisation problems can be categorised as strategic, tactical or operational, depending on the timescale they are dealing with [14]. Strategic problems deal with long-term planning. Tactical problems, such as the one covered here, typically cover a medium-term timescale of a few weeks to a year. While operational problems work on a short-term basis. As mentioned above, the goal of the problem studied in this paper is to define working areas, or *operational units*, consisting of a number of service delivery resources and a set of service distribution points, in a large infrastructure network. In general, service distribution points connect both domestic and commercial properties to key infrastructure services such as gas, water, electricity or telecommunications. Each service distribution point generates demand for service, requiring one or more tasks to be completed at that location. The process of defining operational units is quite complex, and requires defining which set of service distribution points belong to any particular unit. In addition to the geographical set of resources, skilled service delivery resources also need to be allocated to the operational unit. In the context of this work, the skilled resources are utility engineers. However, other examples of such resources include delivery personnel, home care assistants and property management staff. The allocation of individual tasks to skilled resources is not a trivial process, as different tasks might require different skills, and each resource has their own particular, finite skill set.

As mentioned in the previous section, the work in this paper is related to an existing type-2 fuzzy logic based simulation model introduced by Starkey et al. [21], used within the strategic organisational design tool for resource planning at BT. Here we will provide a brief overview of this model, for a more detailed description of the simulation model used, we refer the interested reader to the original paper [21]. This simulation model takes into account a wide variety of uncertainties present in the real-world problem, such as uncertainties in the availability of skilled engineers, uncertain travel and job completion times etc. Taking a service distribution point for each operational unit as input, representing the centre points from which each operational unit is based, the model clusters the rest of the service distribution points in the region into operational units based on the these points. A one-day simulation is then executed, calculating the cost of this allocation for different objectives. Although a number of objectives were introduced for this problem previously [21], here we will use the two objectives considered most important:

- Maximise coverage: the number of tasks that are expected to be completed with this allocation of operational units. This is reported as the percentage of tasks completed, with a higher value preferable.
- Minimise travel distance: the amount of distance engineers cover during the simulation. This objective is in direct conflict with the first objective as usually an engineer will be required to travel between tasks. This is calculated as the average distance travelled by all engineers in the region and is reported in kilometres.

There are a number of challenges that this model must overcome to accurately represent the real-world problem under consideration. The simulation model consists of three main phases. Firstly a set of valid operational units are built based on the input received, each consisting of a number of service distribution points. Once this is done, teams of engineers are allocated to each operational unit. Finally, tasks from each service distribution point are allocated to engineers within each operational unit.

Building the set of operational units. In order to group the service distribution points into operational units, an iterative process is performed. Starting with the set of service distribution points representing operational unit centre points received as input, at each step the closest service distribution points to an operational unit are considered to be added using fuzzy logic rules. When building operational units there are a large number of restrictions in place, preventing certain service distribution points from being allocated to the same operational unit as others. Service distribution points in the same operational unit should not be split by geographical obstacles such as rivers, or by another operational unit (i.e. operational units should be connected). It is also the case that not all service distribution points are the same size, with some service distribution points generating considerably more tasks than others. A membership function based on fuzzified values for the amount of work that a service distribution point generates and the amount of work in the current operational unit is used, ranking each of the service distribution points based on their output defuzzified values. This membership function is designed in a way that aims to balance workload across operational units, with service distribution points generating a large number of tasks more likely to be added to operational units currently containing low volumes of work. The service distribution point with the highest value is added to the operational unit, and removed from the pool of service distribution points to be allocated. This process continues until all service distribution points have been allocated to an operational unit.

Allocating engineers to operational units. In the first instance, each engineer is added to the operational unit closest to their home location. This typically leads to unbalanced teams, with operational units covering areas with a high population density having too many engineers, and rural operational units with low population density having too few. Again an iterative process is performed, reassigning engineers to improve the overall balance of teams. This is done through a bidding process, with operational units that have insufficient resources 'bidding' for the excess engineers from other operational units based on their proximity and skill set. The value of a bid consists of the distance of the 'excess' engineer from the operational unit placing the bid, the strength of demand for their skill set, and the resource deficit of the operational unit. Depending on the current status of each operational unit, different strength bids can be made, with the most under-resourced operational units able to make the strongest bids for engineers from overpopulated operational units. In each case, the operational unit with the highest bid wins and that engineer

will be allocated to that operational unit. As the resource level of an operational unit starts to balance out, its ability to place higher bids is reduced, allowing other underpopulated operational units to successfully bid for additional resources. Upon completion of the bidding process, the set of engineers are reasonably well-balanced across the set of operational units.

Allocating tasks to engineers. Finally, tasks from each service distribution point need to be allocated to engineers within each operational unit. This is done by performing a simulation of a working day, the results of which will be returned as an estimate of the overall quality of the current solution. A fuzzy logic function, inspired by previous work by Mohamed et al. [16], is used to allocate tasks to engineers based on fuzzified values of the distances between the engineers and service distribution points, and the amount of work remaining. Each engineer is assigned tasks until either there is no time remaining in their working day, or there are no more tasks available at service distribution points in their operational unit. Based on these task allocations, which constitute a one-day simulation of a solution to the problem, the proportion of tasks completed (i.e. coverage) and average travel distance can be calculated.

The work in this paper focuses on two key aspects of the optimisation problem under consideration: the number of operational units each region is to be split into, and the choice of centre points for each of these operational units. The optimisation problem at hand requires finding a set of service distribution points, representing the central point of each operational unit, that maximise the volume of work completed (coverage) and minimise the overall travel distance of engineers, within the existing simulation model described above. This is done over a search space of real-valued parameters, which are mapped to a service distribution point ID that acts as the centre of each operational unit. The simulation model builds the operational units from these centre points and allocates work to each engineer. The number of operational units that a region is to be split into must be decided *a priori* to the search process and has a significant impact on the size of the search space and the overall runtime for different instances of the problem. These issues will be discussed in detail in Section 4.

3. Multiobjective optimisation problems and evolutionary algorithms

When dealing with more than one objective, one option is to transform the problem into a single objective problem by linearly combining the objective function values. Although a weighted sum can be used to reflect the relative importance of each component of the objective function based on expert opinion, insight is lost in terms of highlighting the trade-offs that exist between multiple objectives. The aim when addressing a multiobjective optimisation problem directly is to find a set of solutions representing the best trade-off that exists between objectives, often referred to as the pareto front [3]. For one solution to *dominate* another, it must obtain at least the same objective function value in all objectives, and a better objective function value for at least one objective. The pareto front contains the set of *non-dominated* solutions found during the search.

Evolutionary algorithms are a set of metaheuristic search methods inspired by the natural process of evolution [6]. Multiobjective evolutionary algorithms, evolutionary algorithms which simultaneously consider multiple objectives, have been applied to a wide variety of problems over the years, and are accepted to be robust general search methods [7, 24]. Evolutionary algorithms are particularly well-suited to multiobjective problems, as the use of a population allows the exploration of different regions of the pareto front in a single run of the algorithm.

Unlike in single objective methods, where solutions can be compared directly using a single objective function value, deciding how to compare the quality of solutions in multiobjective optimisation problems is not trivial. In order to facilitate the ability to compare solutions, a multiobjective evolutionary algorithm quantifies the value a solution adds to a set of solutions, based on the multiple objective values it yields [3]. The method of doing this depends on the multiobjective evolutionary algorithm used, with each balancing intensification and diversification of the search process in a different manner. For a comprehensive introduction to multiobjective evolutionary algorithms, we refer the interested reader to the book written by Coello Coello et al. [3].

3.1. Multiobjective evolutionary algorithms used in this paper

In our experimentation, we will use four different multiobjective evolutionary algorithms, each described briefly in the following subsections. Previous work for the problem introduced in Section 2 used NSGA-II [4], a pareto dominance-based method, which ranks individuals by pareto dominance when forming each subsequent generation [21]. In addition to NSGA-II, we will also use an indicator-based method (IBEA [25]), a decompositon-based method (MOEA/D [23]) and a second pareto-based multiobjective evolutionary algorithm (SPEA2 [26]).

3.1.1. NSGA-II

The non-dominated sorting genetic algorithm-II (NSGA-II) [4] is one of the most widely used multiobjective evolutionary algorithms in the literature. NSGA-II uses a mechanism which ranks solutions based on the number of solutions they are dominated by. This ranking is used when selecting which solutions to use for mutation and recombination and when deciding which solutions to keep for the following generation. Each individual in the population is given a score, denoting the number of other individuals in the population that dominate it, with a lower score indicating a higher quality solution. Solutions on the pareto front will be dominated by no other solutions and therefore have a score of zero. An explicit mechanism to preserve diversity among the population and promote search across the entire pareto front, known as the *crowding distance*, is also included. Crowding distance is effectively a measure of the density of solutions surrounding a particular solution. In the case that a choice exists between solutions on the same pareto front to carry forward to the next generation, solutions which are in less dense regions of the objective space are preferred.

3.1.2. IBEA

The indicator-based evolutionary algorithm (IBEA) was proposed in 2004 and was shown to be able to outperform NSGA-II and SPEA2 on a number of benchmarks [25]. Rather than using pareto dominance, IBEA uses a given performance measure to rank solutions. Here we use one of the performance measures from the original paper of Zitzler and Künzli [25], the hypervolume measure. Hypervolume measures the amount of objective function space dominated by a particular solution, with respect to a given reference point. IBEA attempts to guide the search process towards the true pareto front by favouring solutions which dominate large regions of the objective space not covered by other solutions. Solutions are ranked in terms of unique hypervolume contribution to the overall hypervolume covered by the population. Selection for each subsequent generation is performed by repeatedly removing the worst solution from the population, then updating the contributions of the remaining solutions, until the desired population size is obtained.

3.1.3. MOEA/D

MOEA/D (Multiobjective evolutionary algorithm based on decomposition) [23] was proposed more recently than the other multiobjective evolutionary algorithms tested in this paper. As the name implies, MOEA/D decomposes a multiobjective problem into a number of scalar subproblems based on an aggregation of all objectives, that are then solved simultaneously by different members of the population. Each subproblem is solved collaboratively, with information shared between subproblems by selecting the solutions for neighbouring subproblems for reproduction more frequently than from elsewhere in the population. A number of variants of MOEA/D exist in the literature. The implementation used in this paper is based on one of the most common variants, derived from the work of Li and Zhang [11] which uses the Tchebycheff approach [15] for decomposition and Differential Evolution [22] for reproduction.

3.1.4. SPEA2

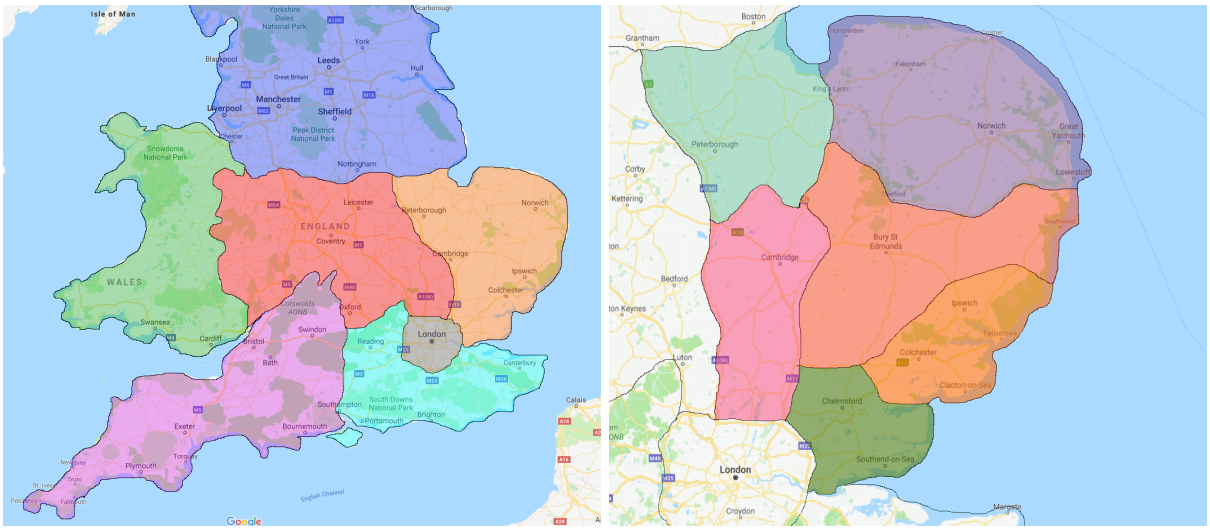
Strength pareto evolutionary algorithm 2 (SPEA2) [26] is an extended version of the SPEA multiobjective evolutionary optimization algorithm. Like NSGA-II, SPEA2 uses pareto dominance to score and rank solutions. In SPEA2, pareto ranking is supplemented by a technique based on the inverse k-th nearest neighbour algorithm to estimate the density of a solution, with individuals in less dense regions of the objective space preferred. SPEA2 maintains a set of non-dominated solutions over time in an external archive, which is updated at each generation. The fitness value of each individual in the current population is calculated, relative to the strengths of the non-dominated solutions in the external archive that dominate it. If at any stage of the search, the external archive exceeds the maximum size permitted, solutions are removed in ascending order of minimum distance to another solution.

4. Analysis of the problem instances studied

For the purposes of the problem introduced in Section 2 the UK is split into 59 *mid-level* regions, each with a different number of engineers and service distribution points, and a variety of geographical constraints present. When an 'instance' is referred to hereafter, this corresponds to one of these 59 regions. The problem under consideration requires splitting these mid-level regions into smaller operational units. Each of these 59 instances are solved separately, using a predetermined number of operational units. Figure 1 (a) shows the highest level of operational unit separation in the UK. Each operational unit at this level would be overseen by a regional manager. These high-level operational units are further split into mid-level operational units as shown in Figure 1 (b), of which there are 59 as mentioned above. Each operational unit at this level would be overseen by an area manager, who would report to their respective regional manager. Finally, Figure 1 (c) gives an example of the lower-level operational units that are to be designed in this work. Each operational unit at this level will have a team and a manager, where the manager would report to the area manager. The X's marked in Figure 1 (c) indicate example service distribution points.

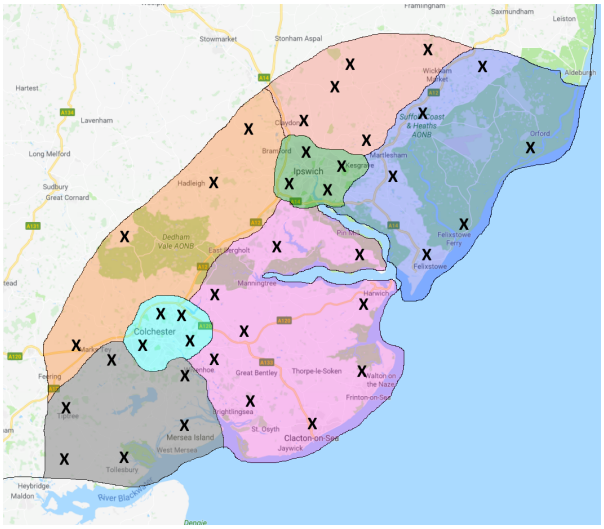
The current practice of the planning team is typically to treat all regions equally in terms of the computational resources allocated to solve each problem. This is not ideal, due to the variety of properties that the different regions have in terms of geographical constraints, number of service distribution points and number of engineers. In the following subsections we will analyse the search space and the execution time of the simulation model for each of the regions in the UK. Some of the considerations that must be made when solving these problems are highlighted, particularly

with respect to the number of operational units that are used, with a subset of instances selected for the main set of experiments that follow in Section 5.



(a) High-level operational units

(b) Example of mid-level operational units



(c) Example of low-level operational units

Figure 1: Overview of the hierarchical operational unit structure in the UK

4.1. Search space size

As introduced in Section 2 above, the simulation model accepts a set of service distribution points as input. These are represented by real-valued parameters, which are then mapped to the ID of a service distribution point location at the centre of each operational unit. The simulation model builds the operational units from these centre points and allocates work to each engineer. Based on a preferred range of 20-24 engineers per operational unit as specified by BT, a 'valid' set of ideal operational units for each instance can be derived from the total number of engineers in that region. The properties of a selection of the 59 instances are given in Table 1.

Table 1: Properties of a subset of instances

Region	Service distribution points	Engineers	Ideal operational unit split(s)
Bristol	75	142	6, 7
Colchester & Ipswich	108	148	7
Crayford	26	267	12, 13
Guildford	52	94	4
Highlands & Islands	316	122	6
Humber	179	201	9, 10
North East Scotland	205	147	7
North London	32	155	7
West Central London	15	41	2
West Yorkshire	105	270	13

Given that the largest number of engineers is 270 and the smallest is 41, the set of possible valid operational units for each instance is between two and thirteen. It is worth mentioning here that there is no direct correlation between the number of service distribution points in an instance and the number of engineers. For example, Highlands & Islands has a very large number of service distribution points with relatively few engineers. However, Crayford has far fewer service distribution points but a very large number of engineers. This is due to the fact that not all service distribution points have the same workload, indeed a large service distribution point in one region of the country might have more work than many smaller service distribution points in another.

Based on the number of operational units and the number of service distribution points, we can compute an estimate of the number of possible solutions. Note that the actual value is slightly less in practice, when invalid states are excluded. When solving a particular instance, for each operational unit, we select a service distribution point to act as a centre point, therefore the number of possible solutions excluding duplicates is approximately:

$$\frac{\#ServiceDistributionPoints!}{(\#ServiceDistributionPoints - \#OperationalUnits)!} \tag{1}$$

Of the 59 instances, there are a number which have a search space that is too small for them to be interesting as search problems. These instances typically have less than 100 engineers, where the number of operational units should be two, three or four. As an example, West Central London has 15 service distribution points and 41 engineers to be split into two operational units. This leads to only 210 $(15!/(15-2)!)$ possible solutions, which could be enumerated exhaustively in a matter of seconds or minutes. At the other end of the scale, the search space for West Yorkshire split into thirteen operational units contains $8.69 * 10^{25}$ solutions $(105!/(105-13)!)$, which is computationally infeasible to enumerate in a reasonable amount of time. This gives an idea of the scope in terms of search space size there is across the problem instances considered.

A question that the planning team can often be asked is whether to recommend increasing or reducing the number of operational units in a particular region. Although this can sometimes mean breaching the 20-24 engineers per unit guideline, there can be managerial and operational reasons for allowing this. Table 2 gives an example of the exponential growth in search space size that occurs as the number of operational units increases for the Colchester & Ipswich region.

This growth must be considered when allocating computational resources to the search process, particularly when trying to compare solutions with a different number of operational units.

Table 2: Estimated number of possible solutions for Colchester & Ipswich with different operational units

Number of operational units	Possible states
1	108
2	11,556
3	1,224,936
4	128,618,280
5	13,376,301,120
6	1,377,759,015,360
7	140,531,419,566,720

Table 2 highlights the additional complexity of the problem when deciding the number of operational units to use, as an instance of each problem would need to be run for each case (for example one for six operational units and one for seven operational units). Thus, it is important to make an operational decision as to how many operational units should be deployed in a region, defining the constraints of the problem, with the planning team tasked with finding the best designs for the specified number of operational units. Having calculated the number of possible states for a region, based on the number of operational units, we are able provide guidance on the complexity of the problem and recommend settings when searching for solutions. One example case would be to run the simulation for every possible state, if the number of states allows this in reasonable time.

4.2. Time taken to run the simulation

In addition to the differences in search space size, preliminary experimentation also indicated that there is also a large variation in the execution time of the model for different regions. Using the standard NSGA-II implementation from MOEA Framework (`http://moeaframework.org/`), we will calculate the time taken to perform 1000 evaluations of the simulation for each of the 59 regions. All experiments are performed on an Intel Core i5-6200U 2.3 GHz CPU with 8 GB memory. Here we fixed the number of operational units to seven for all regions, as this is the most frequent 'ideal' number of operational units based on the 20-24 engineers per unit guidelines. It is also halfway in terms of size between the maximum and minimum valid operational unit sizes (two and thirteen) over all regions. Results are reported in seconds as an average of ten runs, and are plotted in Figure 2 with respect to the number of service distribution points in the region.

The range of execution times between regions is particularly striking, with a clear correlation existing between the number of service distribution points in the region and the time taken to run the simulation. The slowest case is the Highlands & Islands region, which takes over 34 seconds to reach 1000 evaluations on average, although this can be considered as an outlying case. This region has over 300 service distribution points with relatively few engineers (122), and a unique geography, covering an extremely large, sparsely populated area in the north of Scotland, and islands including the Outer Hebrides and Shetland Islands. Aside from this region, there are three regions that require just over 24 seconds on average to complete 1000 evaluations. At the other end of the scale, there are three regions that only require around 0.3 seconds to perform 1000 evaluations. This disparity has huge repercussions on the scalability of solving different instances

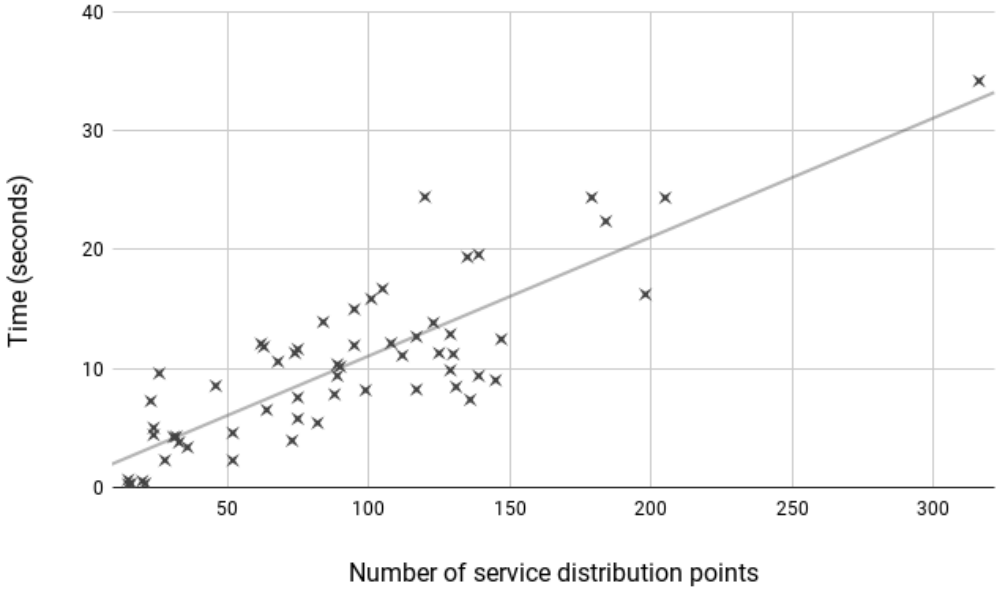


Figure 2: Time (in seconds) for NSGA-II to evaluate 1000 solutions for each of the 59 regions

of this problem. The median time taken to perform 1000 evaluations across all 59 instances is 9.58 seconds.

As the ideal number of operational units is not always seven for all regions, we also tested the amount of time required for NSGA-II to perform 1000 evaluations over the entire range of ideal operational unit sizes, for a smaller subset of instances. Here we have chosen four instances across the spectrum of runtimes observed in Figure 2: Guildford (2.26 seconds), Bristol (5.76 seconds), Crayford (9.58 seconds) and Humber (24.37 seconds). Figure 3 gives the time taken for NSGA-II to evaluate 1000 solutions for each of these instances, for two to thirteen operational units, as an average over 10 runs.

This is also interesting as there is a clear link between the number of operational units used and the execution time of the simulation. In general, as the number of operational units decreases, the time to perform 1000 evaluations increases. However, the rate of increase is different for different regions. This highlights one of the idiosyncrasies of the underlying model. When allocating work to a particular engineer, there are a set of possible service distribution point locations that each engineer can visit. As the operational units introduce geographical constraints on the working areas of engineers, if there are more operational units, and hence fewer service distribution points per unit, there are fewer possibilities to consider when allocating work to a particular engineer, leading to less computational expense.

From this preliminary analysis, the key consideration to be made from a planning perspective is that increasing the number of operational units has the dual effect of vastly increasing the search space, whilst slightly reducing the execution time of each simulation run. This is particularly pertinent, as the planning team may potentially consider increasing the number of operational units in a given region, and must consider the effect this will have on the time that should be allocated to finding solutions to the problem. It is important to note that this does not provide

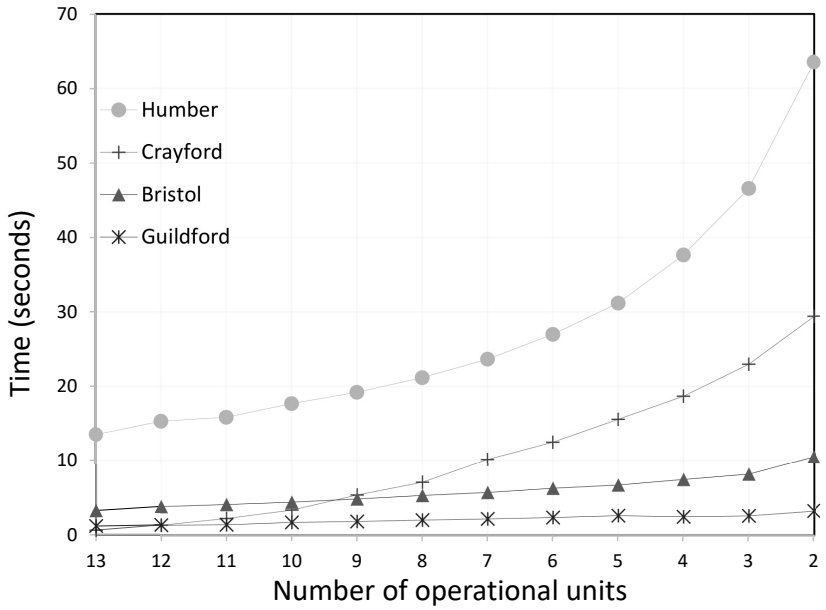
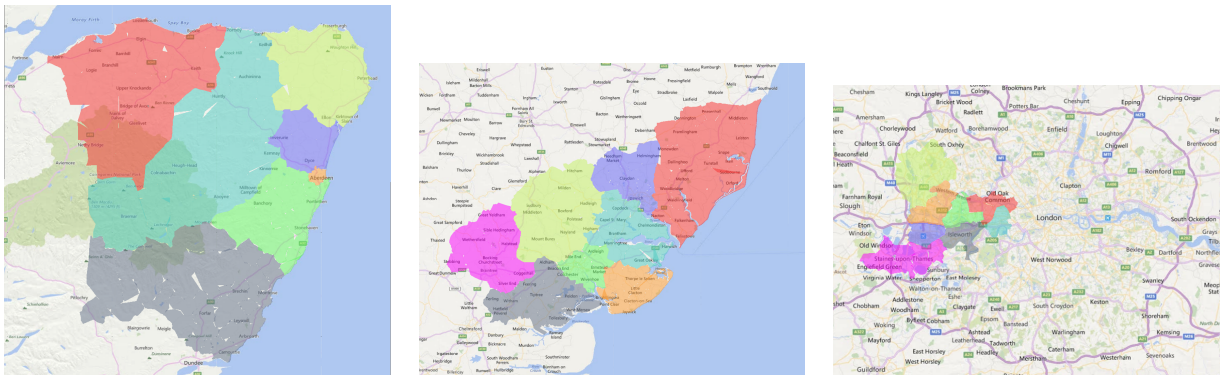


Figure 3: Time (in seconds) for NSGA-II to evaluate 1000 solutions for different operational unit sizes

any insight into the relative difficulty of solving a particular problem.

4.3. Problem instances considered in detail

Here we choose three of the 59 regions to study in detail in the experiments in Section 5. All three regions have a similar number of engineers. However, they differ in terms of the number of service distribution points and geographical spread of service distribution points over the region. Figure 4 shows the three regions used, North East Scotland, Colchester & Ipswich and North London, which represent a low, medium and high density of service distribution points respectively. The shaded areas overlaid onto each map give sample operational units for each area. North East Scotland has 205 service distribution points and 147 engineers, Colchester & Ipswich has 108 service distribution points and 148 engineers and North London has 32 service distribution points and 155 engineers. As BT require a total of 20-24 engineers per operational unit, they are all problems for which a solution with seven operational units is preferred. In the experiments presented in Section 4.2, the time taken for NSGA-II to perform 1000 evaluations for each of these regions was 24.35, 12.11 and 4.26 seconds respectively.



(a) North East Scotland (b) Colchester & Ipswich (c) North London

Figure 4: Low, medium and high density areas used

5. Computational Results

This section presents a set of experiments to analyse different aspects of designing the working areas for maintenance engineers. Firstly, we consider the effect that using a different number of operational units has on the pareto fronts obtained. Secondly, the effect of using different multiobjective evolutionary algorithms across different regions is examined, applying the four multiobjective evolutionary algorithms introduced in Section 3 to the three instances discussed in Section 4.3. Finally we look in detail at the progress made by each multiobjective evolutionary algorithm during a run.

As in the previous section, all experiments are performed on an Intel Core i5-6200U 2.3 GHz CPU with 8 GB memory. The implementations for all four multiobjective evolutionary algorithms are taken from MOEA Framework using the default settings. All four methods use Polynomial Mutation [1] with probability $p_m = 1/N$, where N is the population size, using a distribution index $\eta_m = 20.0$. NSGA-II, IBEA and SPEA2 use Simulated Binary Crossover (SBX) [1] with probability $p_c = 1.0$ and distribution index $\eta_c = 15.0$. The version of MOEA/D is the well-known variant from Li and Zhang [11], which uses Differential Evolution, with a crossover rate of 0.1 and step size = 0.5. For MOEA/D the number of subproblems in the neighborhood of each subproblem $T = 0.1N$, the probability of mating with solutions from within the neighborhood $\delta = 0.9$, and the maximum number of solutions that an offspring can replace $\eta_r = 0.01N$.

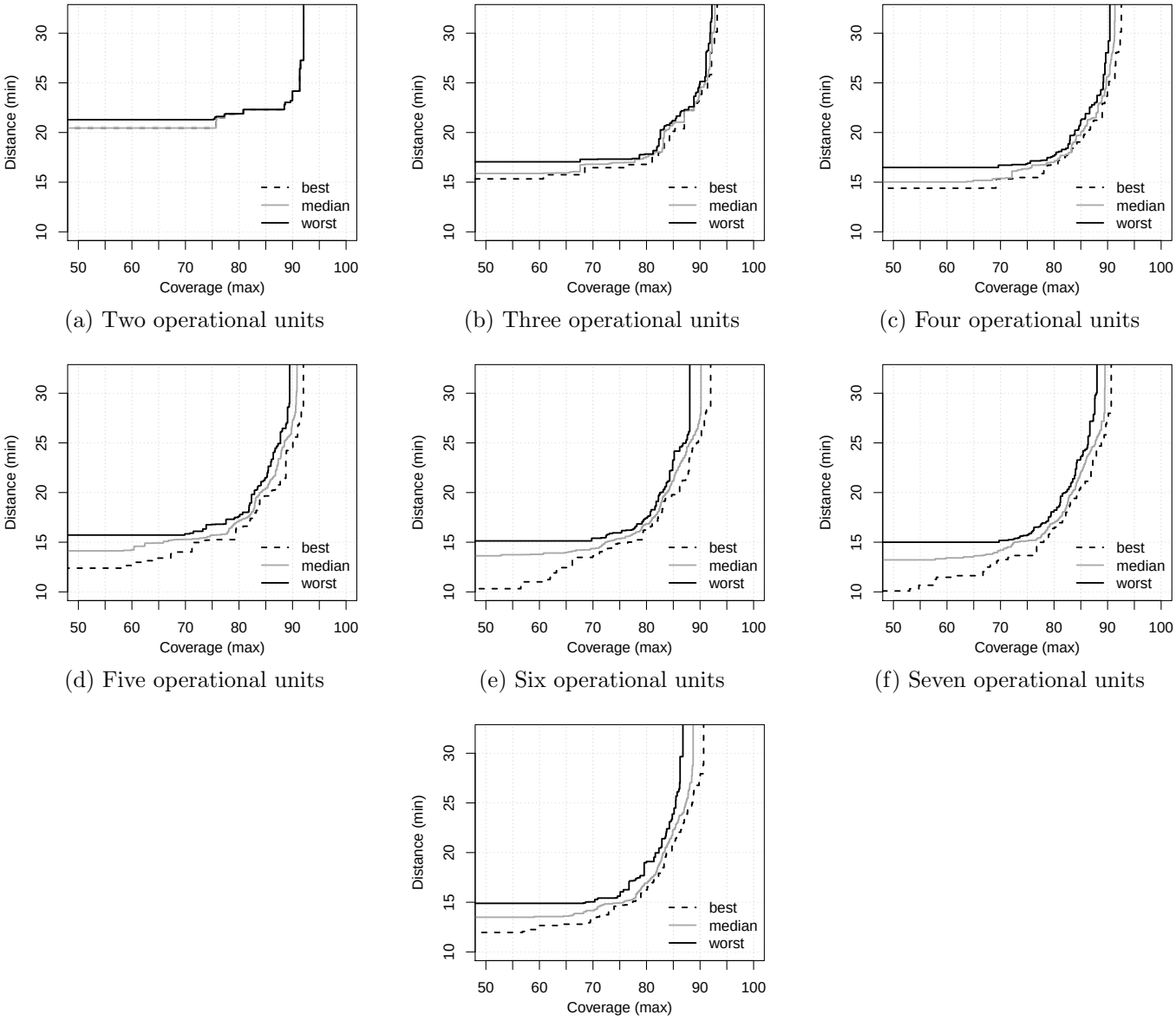
Each multiobjective evolutionary algorithm has a population size of 100, and terminates after 50,000 evaluations. Based on the preliminary experimentation in Section 4.2, this corresponds to a runtime of 8 minutes for the median of the 59 instances and 20 minutes in the worst case. These runtimes are considered acceptable by the planning team at BT. Experiments are repeated 30 times for each multiobjective evolutionary algorithm on each instance.

5.1. Assessing the effect changing the number of operational units has on pareto fronts

The choice of how many operational units to split a region into is a critical factor when the planning team is tackling this problem. Section 4.1 discussed the effect that changing the number of operational units to find a solution for has on the search space, highlighting that increasing the number of operational units vastly increases the size of the search space. Section 4.2 showed a reduction in runtime of the model, effectively the cost of computing the fitness function, when the

number of operational units is increased. In this subsection, we will investigate the impact that changing the number of operational units has on the pareto fronts obtained by a multiobjective evolutionary algorithm, using NSGA-II as an illustrative example.

Figure 5 shows the empirical attainment function (EAF) plots for 30 runs of NSGA-II, with 50,000 evaluations, using two to eight operational units for the Colchester & Ipswich instance. EAF plots provide an overview of the attainment surfaces of methods solving multiobjective optimisation problems. The attainment surfaces of the best and worst end-of-run solutions found over the 30 runs are plotted directly, with the median pareto front showing the area which is attained in over 50% of runs. Note, the objectives are to maximise coverage on the x-axis and minimise average distance on the y-axis.



(a) Two operational units

(b) Three operational units

(c) Four operational units

(d) Five operational units

(e) Six operational units

(f) Seven operational units

(g) Eight operational units

Figure 5: Empirical attainment function plots for NSGA-II applied to the Colchester & Ipswich instance with different numbers of operational units

A clear pattern emerges with a visible shift in the pareto fronts, showing the relationship between the number of operational units, and the objective function values obtained for both distance and coverage. In general, as the number of operational units increases, the average travel distance decreases. However, this is at the expense of coverage which also decreases. Observe in Figure 5a that for two operational units, the lowest distance is above 20km, and a number of solutions are found providing over 90% coverage. When the number of operational units is increased

to eight, as in Figure 5g, the lowest distance is down to below 15km, but very few solutions are found with over 90% coverage. This is not a surprise when we consider the underlying model. An increased number of operational units places more restrictions on the service distribution points that each engineer can work on, creating 'invisible' boundaries that engineers are not able to cross. This is an important point to consider when the planning team is comparing solutions with a different number of operational units. It is not reasonable to expect a solution with more operational units to reduce distance or increase coverage compared to one with fewer operational units.

Figure 5 also shows a trend of decreased consistency in terms of the pareto fronts obtained, as the number of operational units increases. A clear gap emerges between the best, median and worst pareto fronts obtained as the number of operational units is increased, indicating a greater variation in performance over the 30 runs. This is likely due to the growth in search space size discussed previously. Given the estimated search sizes shown in Table 2, it is possible to sample all two operational unit solutions for this instance in the 50,000 evaluations allowed. This explains the similarity between the best, worst and median pareto fronts over 30 runs in Figure 5a, where it is difficult to distinguish between the three lines on the graph.

5.2. Results of using different multiobjective evolutionary algorithms for different regions

As discussed previously, existing work on this problem focused on using NSGA-II to search for the centre points from which the simulation builds operational units [21]. Note that the 50,000 evaluations per run used here is significantly more than the 800 used in previous work. Here we perform experiments to compare NSGA-II to IBEA, MOEA/D and SPEA2 over the three problem instances introduced in Section 4.3. Table 3 shows the minimum, median and maximum hypervolumes, a typical measure used for performance in multiobjective optimisation problems, for each of the four multiobjective evolutionary algorithms over all three instances, using seven operational units. The hypervolume for a particular multiobjective evolutionary algorithm is the volume of objective space dominated by the final set of solutions found, with a larger value indicating superior performance. The relative rank of each multiobjective evolutionary algorithm is also included in brackets for comparison.

Based on the raw data in the table, overall it appears that NSGA-II is outperforming all other methods in the case of minimum, maximum and median hypervolume obtained, with MOEA/D performing badly. The results of a Kruskal-Wallis test within a 95% confidence interval indicate that not all of these differences are statistically significant. Although NSGA-II performs best on average, for Colchester & Ipswich there is no statistically significant difference between the results of NSGA-II and SPEA2. In the case of North East Scotland there is no statistically significant difference between NSGA-II and MOEA/D. The results for North London are more clear cut, with NSGA-II outperforming all other methods on average, with a statistically significant difference to the results obtained by the other multiobjective evolutionary algorithms.

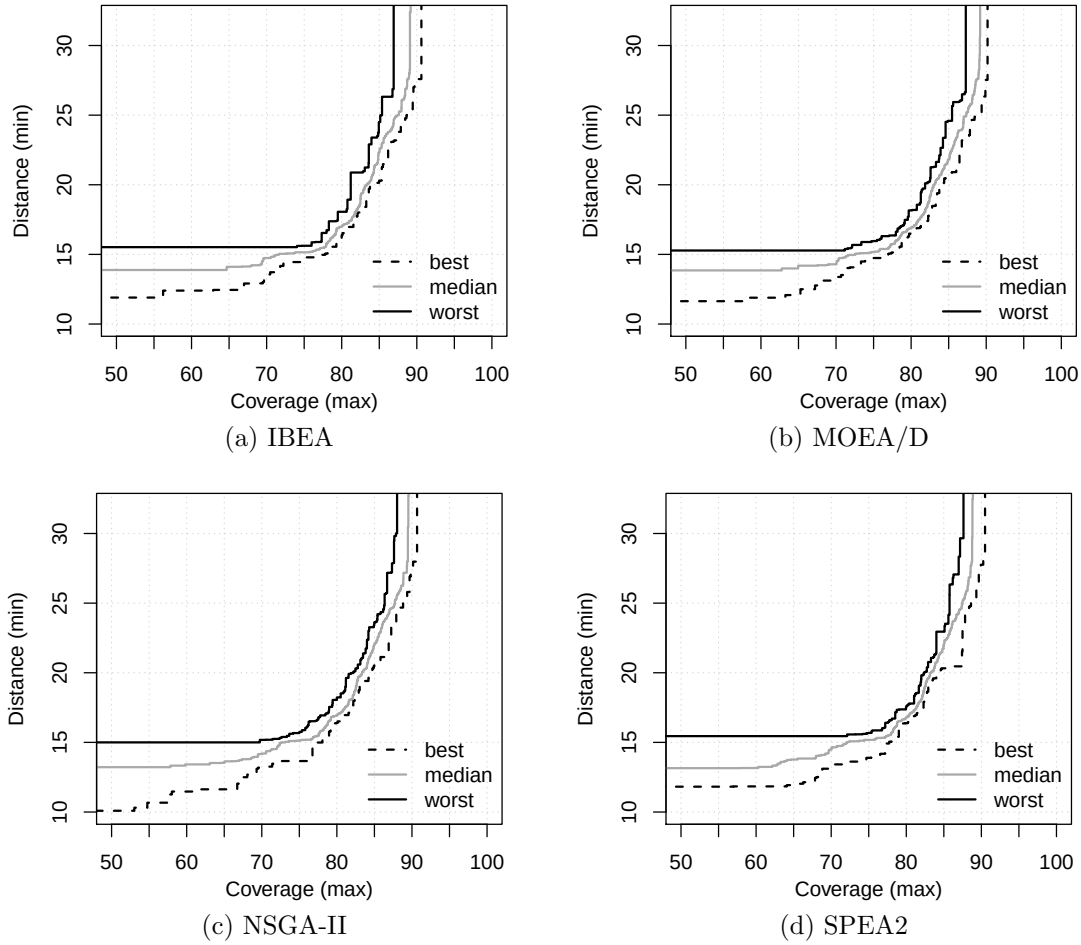
Looking more closely at some of the pareto fronts obtained, Figure 6 shows the EAF plots for each of the four multiobjective evolutionary algorithms, based on the pareto fronts found at the end of each run. These plots show the performance over 30 runs of the Colchester & Ipswich instance with seven operational units.

One observation that can be made from these figures is that the pareto fronts for NSGA-II are far more smooth than those of the other three multiobjective evolutionary algorithms. This suggests that its performance is more consistent across multiple runs, and that the solutions found are well spread across the pareto front. The 'jagged' nature of the EAFs for IBEA suggests that

Table 3: Hypervolumes and relative ranks of 4 multiobjective evolutionary algorithms on 3 instances of the problem

	Hypervolume			
	NSGA-II	MOEA/D	IBEA	SPEA2
	Colchester & Ipswich			
Min	0.608 (1)	0.574 (4)	0.579 (2)	0.575 (3)
Median	0.643 (1)	0.620 (4)	0.625 (3)	0.638 (2)
Max	0.681 (2)	0.675 (3)	0.660 (4)	0.697 (1)
	North East Scotland			
Min	0.667 (1)	0.624 (3)	0.616 (4)	0.644 (2)
Median	0.688 (1)	0.684 (2)	0.677 (4)	0.683 (3)
Max	0.717 (2)	0.721 (1)	0.716 (3)	0.706 (4)
	North London			
Min	0.606 (1)	0.586 (4)	0.599 (3)	0.602 (2)
Median	0.636 (1)	0.610 (4)	0.615 (3)	0.626 (2)
Max	0.654 (1)	0.634 (3)	0.631 (4)	0.645 (2)
Average rank	(1.22)	(3.11)	(3.33)	(2.33)

this method becomes trapped in certain regions of the search space, and is unable to diversify effectively over the entire pareto front. This is in line with observations made in previous work [13]. The crowding distance measure in NSGA-II works to overcome this problem, promoting diversity across the pareto front when selecting the individuals that make up each generation. In general, the 'worst' plots for NSGA-II are much better than those for the other two methods. It is difficult to draw general conclusions from the plots of best solutions found as these solutions are potentially outliers, as can be seen in Figure 6c for SPEA2. Although only the figures for Colchester & Ipswich are provided here, similar behaviour was observed in the EAFs for both North East Scotland and North London.



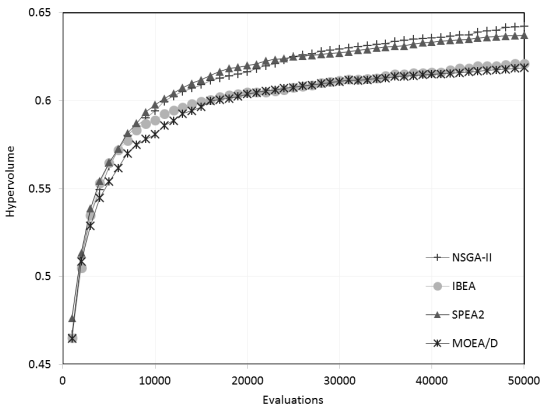
(a) IBEA (b) MOEA/D

(c) NSGA-II (d) SPEA2

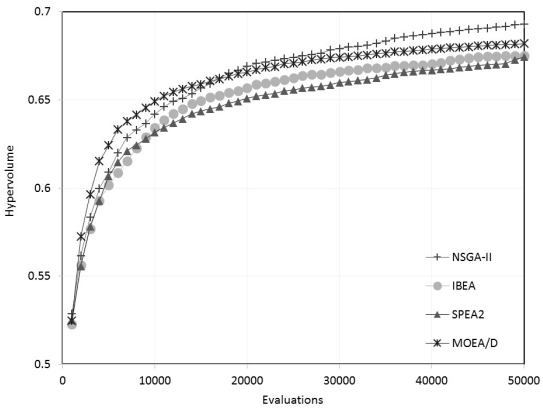
Figure 6: Empirical attainment function plots for each of the multiobjective evolutionary algorithms applied to Colchester & Ipswich with seven operational units

5.3. Progress of multiobjective evolutionary algorithms during a run

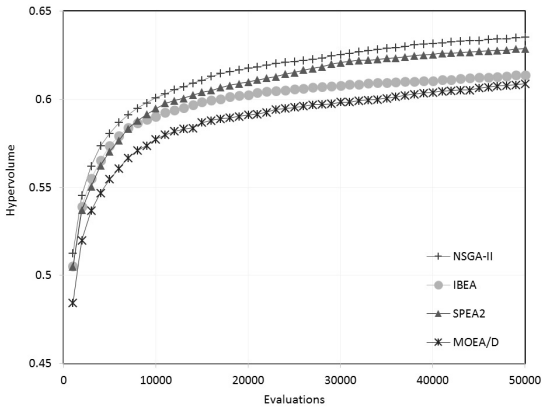
Figure 7 plots the average hypervolume, taken at 1000 iteration intervals, for each of the four multiobjective evolutionary algorithms applied to Colchester & Ipswich, North East Scotland and North London in the previous subsection.



(a) Colchester & Ipswich



(b) North East Scotland



(c) North London

Figure 7: Average hypervolume over time for 30 runs of each multiobjective evolutionary algorithm

From these figures, we see that overall NSGA-II is performing very well, typically achieving good results in terms of hypervolume in the early stages of a run, in addition to outperforming the other three multiobjective evolutionary algorithms at the end. For the Colchester & Ipswich and North London instances, the two pareto dominance-based multiobjective evolutionary algorithms (NSGA-II and SPEA2) are clearly outperforming the other two methods. However this is not the case in North East Scotland. Although SPEA2 performs well in the early stages and most of the way through the runs for Colchester & Ipswich, and reasonably well in North London, it is the worst performing method for North East Scotland. MOEA/D is the worst performing multiobjective evolutionary algorithm for both Colchester & Ipswich and North London, but is performing relatively well on the North East Scotland instance, particularly in the early stages. In general IBEA performs poorly on all three instances in terms of final hypervolume. However, in the early stages for Colchester & Ipswich and particularly North London, it starts off improving very quickly but stagnates, making very flat progress in the latter stages of runs. As observed previously, IBEA can struggle to maintain diversity among solutions. In the case of North London, as this instance is very dense, many service distribution points are located in close proximity to one another. As a result, many different solutions might give similar results, leading to plateaus in the search space that IBEA cannot move away from. For all four evolutionary algorithms, these

graphs seem to indicate that there are still noticeable improvements in solution quality obtained towards the end of runs. It is worth noting that at this point in the search, it is often in the extreme points of the objective space that improvement is found in hypervolume terms. As can be seen in Figure 6 in the section above, some of the best found solutions on the pareto front obtain very low distance values, however an unacceptably small percentage of work is actually being completed, so such solutions are of limited use to the company.

Table 4 shows the average execution times over 30 runs for each of the four multiobjective evolutionary algorithms on each of the three problem instances.

Table 4: Average execution time in seconds for different multiobjective evolutionary algorithms on Colchester & Ipswich, North East Scotland and North London instances

Instance	NSGA-II	IBEA	MOEA/D	SPEA2
Colchester & Ipswich	744.04	715.83	784.91	914.09
North East Scotland	1054.99	954.03	1022.89	1083.10
North London	236.15	135.76	205.19	207.78

Although we expect a difference in runtime over different instances based on the results in Section 4.2, this table also highlights the differences between different multiobjective evolutionary algorithms on the same instance. In all cases IBEA is the quickest method. However, as we have seen above, this comes at the cost of performance. SPEA2 is particularly slow on the Colchester & Ipswich and North East Scotland instances but is the second fastest for North London. MOEA/D is slower than NSGA-II for Colchester & Ipswich, but quicker in the case of the other two instances.

6. Conclusion

In this paper we have presented a series of experiments, analysing the considerations that need to be made when addressing a real-world multiobjective workforce optimisation problem. This work has yielded important practical insights which are now used to guide real-world operational planning decisions, highlighting the issues that must be considered when solving this problem. Analysis of the problem showed the wide variety in terms of search space and cost of fitness evaluation of seemingly similar problem instances. These are important factors to take into account when trying to solve complex problems in the real world. Our experimentation analysed the effect of a number of decisions that must be made *a priori* to the search, including the number of operational units that the region is to be split into and the multiobjective evolutionary algorithm used. Results showed that a clear relationship exists between the two competing objectives and the choice of number of operational units. The number of operational units has a profound effect on the objective function values that can be expected from an multiobjective evolutionary algorithm solving the problem. Previous work on this problem used NSGA-II fairly arbitrarily, as it is a particularly well-known multiobjective evolutionary algorithm. The empirical results observed here confirm that this was a reasonable choice to make, with NSGA-II performing well compared to the multiobjective evolutionary algorithms tested. These observations have provided key insights for the planning team using this model in the real world, and form the basis of a new best practice guide for using their strategic planning tool.

The work in this paper considered two core objectives, maximising the total volume of work completed, whilst minimising the total travel distance of engineers. A number of other objectives

could be considered, either from a management perspective, e.g. minimising the range of operational unit sizes, or from an engineer's perspective, e.g. minimising the maximum travel distance of a single engineer. Future work will include more objectives such as these, analysing the interplay between objectives. When the number of objectives increases, so-called *many-objective* optimisation problems, not all multiobjective evolutionary algorithms scale well to solve such problems. The performance of the multiobjective evolutionary algorithms used here will be compared to other methods designed specifically for many-objective problems.

Hyper-heuristics are high-level search methodologies which operate over a search space of heuristics rather than on solutions directly [2]. Hyper-heuristic multiobjective evolutionary algorithms, utilising multiple mutation and crossover operators, or even multiple multiobjective evolutionary algorithms, have shown to be successful in a number of multiobjective optimisation problems [12, 9]. Our ongoing research is now focusing on combining different operators and multiobjective evolutionary algorithms, utilising the strengths of each during the search process.

Acknowledgments

This work has been partially funded by the DAASE project, EPSRC programme grant EP/J017515/1.

References

[1] Agrawal, R. B., 1995. Simulated binary crossover for continuous search space. Complex systems 9 (2), 115–148.
[2] Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J. R., 2010. A classification of hyper-heuristic approaches. In: Handbook of metaheuristics. Springer, pp. 449–468.
[3] Coello, C. A. C., Lamont, G. B., Veldhuizen, D. A. V., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd Edition. Springer Science.
[4] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.
[5] Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C., Vansteenwegen, P., 2013. Heuristics for the traveling repairman problem with profits. Computers & Operations Research 40 (7), 1700–1707.
[6] Eiben, A. E., Smith, J. E., 2003. Introduction to evolutionary computing. Springer.
[7] Fonseca, C. M., Fleming, P. J., 1995. An overview of evolutionary algorithms in multiobjective optimization. Evolutionary computation 3 (1), 1–16.
[8] Gorman, M. F., Kanet, J. J., 2010. Formulation and solution approaches to the rail maintenance production gang scheduling problem. Journal of Transportation Engineering 136 (8), 701–708.
[9] Guizzo, G., Bazargani, M., Paixao, M., Drake, J. H., 2017. A hyper-heuristic for multi-objective integration and test ordering in google guava. In: Search Based Software Engineering - 9th International Symposium (SSBSE 2017). Vol. 10452 of LNCS. Springer, pp. 168–174.
[10] Karnik, N. N., Mendel, J. M., Liang, Q., 1999. Type-2 fuzzy logic systems. IEEE Transactions on Fuzzy Systems 7 (6), 643–658.
[11] Li, H., Zhang, Q., 2009. Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation 13 (2), 284–302.
[12] Li, W., Özcan, E., John, R., 2017. Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation. Renewable Energy 105, 473–482.
[13] Li, W., Özcan, E., John, R., Drake, J. H., Neumann, A., Wagner, M., 2017. A modified indicator-based evolutionary algorithm (mIBEA). In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2017). pp. 1047–1054.
[14] Lidén, T., 2015. Railway infrastructure maintenance-a survey of planning problems and conducted research. Transportation Research Procedia 10, 574–583.
[15] Miettinen, K., 1998. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers.
[16] Mohamed, A., Hagras, H., Shakya, S., Owusu, G., 2013. A fuzzy-genetic tactical resource planner for workforce allocation. In: IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS 2013). IEEE, pp. 98–105.

[17] Peng, F., Ouyang, Y., 2014. Optimal clustering of railroad track maintenance jobs. Computer-Aided Civil and Infrastructure Engineering 29 (4), 235–247.
[18] Pour, S. M., Drake, J. H., Burke, E. K., 2018. A choice function hyper-heuristic framework for the allocation of maintenance tasks in danish railways. Computers & Operations Research 93, 15–26.
[19] Salehipour, A., Sörensen, K., Goos, P., Bräysy, O., 2011. Efficient GRASP+ VND and GRASP+ VNS metaheuristics for the traveling repairman problem. 4OR: A Quarterly Journal of Operations Research 9 (2), 189–209.
[20] Starkey, A., Hagras, H., Shakya, S., Owusu, G., 2016. A comparison of particle swarm optimization and genetic algorithms for a multi-objective type-2 fuzzy logic based system for the optimal allocation of mobile field engineers. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2016). IEEE, pp. 5068–5075.
[21] Starkey, A., Hagras, H., Shakya, S., Owusu, G., 2016. A multi-objective genetic type-2 fuzzy logic based system for mobile field workforce area optimization. Information Sciences 329, 390–411.
[22] Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11 (4), 341–359.
[23] Zhang, Q., Li, H., 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11 (6), 712–731.
[24] Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., Zhang, Q., 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation 1 (1), 32–49.
[25] Zitzler, E., Künzli, S., 2004. Indicator-based selection in multiobjective search. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 832–842.
[26] Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: Improving the strength pareto evolutionary algorithm. In: Proc. EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control With Applications to Industrial Problems.