Dynamic optimization of preventative and corrective maintenance schedules for a large scale urban drainage system

Yujie Chen[a,b,*], Peter Cowling[a,b], Fiona Polack[a,b], Stephen Remde[a,b,c], Philip Mourdjis[a,b]

[a] *York Centre for Complex Systems Analysis, University of York, United Kingdom*
[b] *Department of Computer Science, University of York, United Kingdom*
[c] *Gaist Solutions Ltd.**, United Kingdom*

Abstract

Gully pots or storm drains are located at the side of roads to provide drainage for surface water. We consider gully pot maintenance as a risk-driven maintenance problem. We explore policies for preventative and corrective maintenance actions, and build optimised routes for maintenance vehicles. Our solutions take the risk impact of gully pot failure and its failure behaviour into account, in the presence of factors such as location, season and current status. The aim is to determine a maintenance policy that can automatically adjust its scheduling strategy in line with changes in the local environment, to minimize the surface flooding risk due to clogged gully pots. We introduce a rolling planning strategy, solved by a hyperheuristic method. Results show the behaviour and strength of the automated adjustment in a range of real-world scenarios.

Keywords: Maintenance, Scheduling, Routing, Large scale optimization

1. Introduction

A gully pot is the part of the storm drain that prevents solids and sediment from flushing into sewers, where they cause blockages in the underground surface water collection infrastructure (Butler et al. (1995)). Regular cleaning is required for gully pots to function effectively (Karlsson and Viklander (2008), Scott (2012)): typical strategies are to clean all gully pots once or twice a year. If gully pots are not cleaned regularly, partial or complete blockages and accelerated deterioration of the gully pots increases the likelihood of surface water flooding. In extreme situations such as intensive rainfall, a clogged drainage system may cause serious property loss (i.e. BBC (2011, 2012), Shieldsgazette (2012), Leylandguardian (2015)).

In the UK, gully pot cleaning is undertaken by local councils, each using its own strategy. Our research focuses on gully pot cleaning for Blackpool, UK, with data from the local council and from consultants, Gaist Solutions Ltd. Blackpool's gully pot maintenance system records 28,149 gullies in an area of about 36.1 km^2. Analysis of real-world gully pot maintenance records shows that season and weather play a critical role: leaf-fall causes many gully pot blockages, whilst strong winds can blow sand or dirt into gully pots causing partial blockages. Historically, reporting of gully pot issues by local residents varies across the seasons and is the lowest in winter, when short daylight and cold weather reduce footfall.

Blackpool local council has two gully cleaning vehicles but only one cleaning team. On any day, the team either takes out the normal cleaning machine, which uses hydrodynamic pressure and a vacuum to loosen and remove solids and liquids from a gully pot (Karlsson and Viklander (2008)), or uses a specialist machine, equipped for fixing broken gully pots. The cleaning team manager estimates that the average time to clean a gully pot is about 5 minutes, whilst the specialist vehicle takes an average of 10 minutes to fix or replace a damaged gully pot excluding travel time.

Each day there is a schedule of gully pots to visit,

*Corresponding author
**http://www.gaist.co.uk/
Email addresses: yc1005@york.ac.uk (Yujie Chen), peter.cowling@york.ac.uk (Peter Cowling), fiona.polack@york.ac.uk (Fiona Polack), stephen.remde@york.ac.uk (Stephen Remde), pjm515@york.ac.uk (Philip Mourdjis)

starting and ending at the depot. Either maintenance vehicle departs the depot at 09:00 and returns no later than 17:00. During servicing, some gully pots are inaccessible, usually due to parked vehicles. If the team encounters a broken gully pot during normal cleaning, it is recorded and subsequently added to the schedule of the specialised vehicle. Scheduling also needs to take account of residents' reports of problematic gully pots: depending on the local risk, these emerging events should be scheduled 5 to 20 days from when they are recorded.

In constructing routes, we take account of both preventative cleaning, and response to emerging events. We would like to generate actual gully pot maintenance schedules that are dynamically optimised to take account of each gully's up-to-date status and its risk impact (which varies across the city). We assume that the capacity of the cleaning vehicle is sufficient for normal daily work, and the waste disposal process is beyond our scheduling programme. Thus, no vehicle capacity constraint is considered in this study. We propose a dynamic short-period scheduling approach, with two component tasks, as follows.

1. Decide which gully pots to serve in the near future, according to the current environment and most recent gully pot condition information.
2. Construct daily cleaning routes that minimise the travelling cost, and maximise the number of gullies cleaned every day.

This problem is similar to the well-known periodic vehicle routing problem (PVRP) (Christofides and Beasley (1984), Hemmelmayr et al. (2009), Gulczynski et al. (2011), Baldacci et al. (2011), Vidal et al. (2012)). Since only one vehicle works each day, it could be specified as a periodic travelling salesman problem (PTSP). However, there are a number of distinguishable characteristics in our gully pot cleaning problem. Firstly, during the planning period, not all gully pots can be cleaned. Secondly, there is no hard constraint of overall cleaning frequency for each gully pot, so, rather than a service pattern, we use a function to estimate a failure rate for each gully pot and identify which pots require servicing. Finally, our guiding principle is to minimize the urban surface water flooding risk caused by clogged gully pots, whilst optimizing the cost of daily cleaning routes.

In this paper, we propose an approach that maintains a set of distance optimized routes evolving with the environmental changes over time. We apply a tabu-based hyperheuristic – binary exponential back off (BEBO) (Remde et al. (2009)) which manages a set of route-adapting and scheduling local search operators to improve the solution iteratively.

The remainder of the paper is organized as follows. Section 2 presents the literature survey relevant to this study. Section 3 defines the model and describes the solution approach in detail. A comprehensive discussion and analysis of drainage system maintenance strategies is given in Section 4. Finally, we present the conclusion and directions for future research in Section 5.

2. Related works

2.1. Preventative maintenance and Corrective maintenance

Maintenance is generally categorised into corrective and preventative maintenance (Duffuaa et al. (2001), Ahmad and Kamaruddin (2012)). Corrective maintenance (CM) usually happens after failures occur. It includes actions such as repair and replacement. Tsang (1995) notes that the consequence of doing only corrective maintenance is a high risk of machine downtime and property loss. Preventative maintenance (PM) is an alternative strategy that reduces these risks. In industry, preventative maintenance typically takes place at regular time intervals, based on experience.

Operational research on PM introduces decision making, based on data analysis, with techniques such as time-based (TBM) (e.g. Scarf and Cavalcante (2010), Wu et al. (2010)) and condition-based maintenance (CBM) (e.g. Carnero Moya (2004), Campos (2009)). TBM can be applied when the failure rate is predictable, whilst CBM is employed where conditions are continuously monitored by sensor or any appropriate indicators. A similar approach, tracking real-time operation information, is also applied in dynamic scheduling (e.g. Cowling and Johansson (2002)). There is little research combining PM and CM strategies: Kenne and Nkeungoue (2008) introduce a PM/CM rate control strategy, obtaining a near-optimal maintenance policy for a manufacturing system.

Our gully pot maintenance problem involves geographically distributed points and a strictly-limited service resource. Therefore, instead of finding an optimal maintenance policy for each individual object, the focus of this research is to produce an

optimal maintenance schedule covering all objects within time and resource constraints.

2.2. Maintenance & on-site service problem modelled as periodic vehicle routing problem

The periodic vehicle routing problem (PVRP) model is widely used, and planning routes for maintenance and on-site service is one of its many applications.

Blakeley et al. (2003) use a multiple-objective PVRP to model PM for real-world elevator and escalator maintenance, which includes periodically checking customers' equipment and reacting to call-outs. Travelling time, workload balancing, visiting time window violation and overworking time are considered in a weighted linear function. A two-stage approach is used: 1) assign all tasks to each technician, based on technician skill sets and the geographical distribution of tasks; 2) solve periodic travelling salesman problem (PTSP) for each technician over a 13-week period.

Jang et al. (2006) implement a very similar two stage approach to solve a problem of routing lottery sales representatives to visit lottery retail locations. In their assignment stage, the k-means clustering method is used.

Related work has also been analysed in remote healthcare services. An et al. (2012) consider the home healthcare problem, which needs to provide periodical services to various patients.

Maya et al. (2012) help an education institution to provide periodical services for disabled children. This problem is considered as a multiple depot PVRP as each teaching assistant starts and ends their journey from home.

Alegre et al. (2007) analyse a real-world periodic pick-up of raw materials problem and modelled it as PVRP. The notable characteristic of this research is the very long planning horizon (90 days) compared to other literature.

Tang et al. (2007) model a geographically distributed equipment maintenance scheduling problem as a multiple tour maximum collection problem with time dependent rewards. The rewards are decided based on manufacturer maintenance interval suggestions. The approach has similarity to PVRP, in that a schedule is produced for a given period (e.g. a week or a month). On the other hand, some significant differences include: not all equipment requires a visit within a planning horizon; the objective is not to minimize the travelling cost but to maximize the reward from completing tasks (i.e fixing or checking a machine).

García et al. (2013) consider a perishable products supply problem for a bakery company. Weekly delivery routes from the depot to distributors are generated. This problem introduces a certain flexibility in the delivery date. The authors introduce a bi-objective model that minimizes the total travelled distance and the total stock over the planning horizon simultaneously. Two meta-heuristic approaches such as linked VNS and NSGA-II (Deb et al. (2002)) are applied to produce good approximations of the Pareto front for this bi-objective problem.

2.3. Solutions of PVRP

To solve PVRP-related problems, two main processes are commonly considered. The first approach (e.g. Alegre et al. (2007)) assigns customers to days according to their service pattern and then solves a VRP for each day. This solution transforms PVRP to a multiple depot VRP (MDVRP). The second approach (e.g. Tang et al. (2007)) is to simplify a PVRP to PTSP by assigning customers to each vehicle/salesman. Routes are then built up and scheduled to days. This second approach is usually used when the service fleet is heterogeneous, or when strong ties exist between specific service personnel and customers.

Baldacci et al. (2011) propose a successful exact algorithm for solving the PVRP. To our knowledge, this paper presents the largest PVRP solved by an exact algorithm, at 199 customers.

Meta-heuristics, which are capable of solving large scale real-world problems, are the most common PVRP solvers in literature. Chao et al. (1995) present a two-stage record-to-record algorithm that constructs solutions using several local moves applied one after another. Cordeau et al. (1997) were the first to use a tabu search heuristic for PVRP. During the search, infeasible solutions are allowed and controlled using an adaptive penalty function. Alegre et al. (2007) apply a scatter search framework (Laguna and Marti (2012)) to solve PVRP. The algorithm is based on a problem of assigning calendars to customers in a periodic vehicle loading problem (Delgado et al. (2005)). Another strong meta-heuristic framework for PVRP, variable neighbourhood search (VNS), is proposed by Hemmelmayr et al. (2009). Pirkwieser and Raidl (2010) add a coarsening and refinement process to VNS, called multilevel VNS for PVRP.

More recently, hybrid meta-heuristics present very competitive results in terms of both solution quality and computational time. Gulczynski et al. (2011) describe an integer programming-based heuristic (IPH): in this approach, the reassignment and daily routing processes are repeatedly applied until little or no improvement is found in the current iteration, when a restart initial solution is generated. Gulczynski et al. (2011) report that IPH out-performs the algorithms proposed by Chao et al. (1995), Cordeau et al. (1997), Alegre et al. (2007), Hemmelmayr et al. (2009). Vidal et al. (2012) propose a hybrid genetic algorithm that combines local search and sophisticated population management strategies to guide the search, an approach shown to perform better than all the above algorithms. Cordeau and Maischberger (2012) combine tabu search and iterated local search to give a competitive, broad exploration of the search space. Rahimi-Vahed et al. (2012) propose a modular heuristic algorithm (MHA) that introduces a reference set to guide exploration and exploitation during the search for solutions minimising the number of vehicle used. In addition, this paper also presents a self-learning mechanism that leads the search to assign better customer visit patterns as the solution evolves.

3. Solution

3.1. Modelling

Here, we clarify our gully pot maintenance problem with its natural features using a mathematical model. A geographically-distributed system has N points that need maintenance over a long period D (e.g. 3 to 5 years). Each point i is associated with a risk impact r_i, which measures the value of this point to its surrounding environment (i.e. critical properties). The failure probability of each point changes over time, and can be estimated by a function $P_i(d)$, which measures the probability that gully i is in a failure state on day d. A subset of points, $M \in N$, is scheduled in the next short maintenance period W (e.g. a week or 2 weeks). Other input parameters include the following:

- T_{max}: the maximum travelling time allowed for each route;
- d_{ij}: distance, in terms of travelling time, from gully i to j: we use actual road distance between points;
- t_i: service time at point i.

The objective is to select a judicious subset of points from N and assign them to days of the following short period, in order to minimize the risk in this period:

$$\sum_{d \in W} \sum_{i=1}^{N} r_i P_i(d) \tag{1}$$

This problem is subject to two constraints: (1) for any route, the total travelling time plus the total servicing time does not exceed T_{max} for that route; (2) each route must start and end at the depot.

3.2. Problem preparation

3.2.1. Estimating the risk impact per gully pot

A potential hazard (i.e. surface water flooding) could be exacerbated by both geographic factors (i.e. elevation, soil type) and social-related factors, which are usually influenced by economic, demographic and building types (Cutter et al. (2003)). A higher risk impact here implies that if a particular gully pot is blocked and floods happen, it results in relatively larger economic and social losses. In other words, we prefer to clean the gully pots with larger impact more frequently to keep them working properly. Co-operating with Blackpool local council, we firstly decide a list of social concerns with awareness of their economic and population influence, as shown in table 1. Then, each gully pot is evaluated by its location and the related social concerns.

Based on the existing data from Blackpool council, social concerns are classified in to three groups: 1) residential property; 2) commercial and industrial areas including local and district centres, business zones, and employment sites; 3) public services including schools, hospitals, doctors and public transport routes. In table 1, the estimated value of each item in group 1 is the average residential house price in Blackpool UK GOV (2015). Group 2 takes account of the footfall and critical building prices for each item. The estimated value of items in group 3 is based on average daily operation costs.

Flooding impact analysis involves large uncertainties. Research has shown historic flooding from different perspectives (Changnon (1999), Thieken et al. (2008), Brouwer and Van Ek (2004), Merz et al. (2004)). We do not expect a precise assessment of impact. Instead, we aim to find values that are able to guide gully pot maintenance actions in

Table 1: Social factor evaluation

Group	**Social Concerns**	**Estimated value**	**Value loss from flooding**	**Risk impact**
1	Residential	£113,000	3%	£34
2	Local center	£1,130,000	5%	£580
	District center	£1,695,000	5%	£870
	Business area	£565,000	5%	£290
	Employment sites	£226,000	5%	£116
3	School	£5,168	4%	£71
	Large hospital	£917,808	4%	£377
	Doctors	£9,178	4%	£73
	Bus route	£220	100%	£37

decision making. Here, we mainly focus on direct economic losses using a damage function which relates to property type and water level. Thieken et al. (2008) propose the impact from a range of flood water levels on different building types. After consulting Blackpool Council and Gaist Solutions Ltd., we decide to focus on the impact of floodwater levels of less than 21 cm. This gives value-loss figures (Table 1, third column) of 5%, 3% and 4% for commercial, residential and public service areas, respectively. For public transport we focus on bus routes, estimating the cost of closing a road section due to surface water flooding.

By analysing Blackpool historic flooding frequency (Blackpool (2009)), the probability of flooding events is used to map the flooding value loss to the daily risk impact per gully pot according to its location (last column of Table 1). We assume that gullies in the same section of a street evenly share the responsibility for the risk impact evaluated in that area. Figure 1 illustrates the geographic distribution of gully pot risk impact in Blackpool.

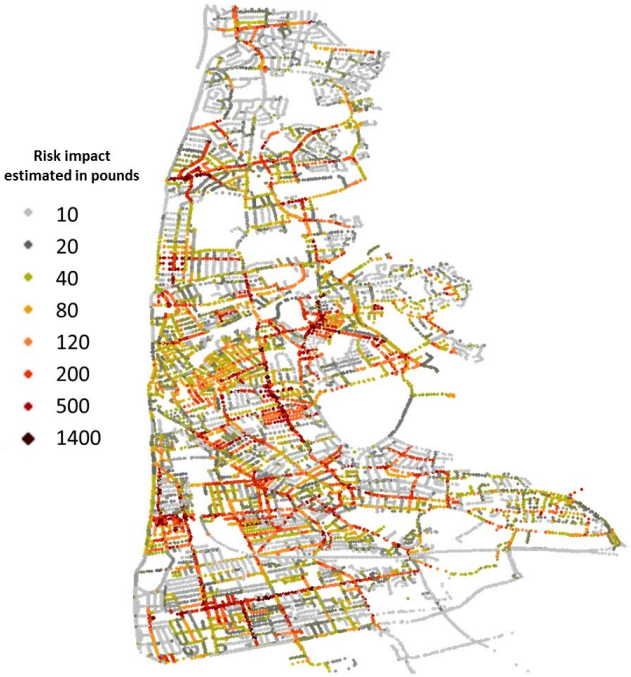


Figure 1: Gully pot risk impact in Blackpool

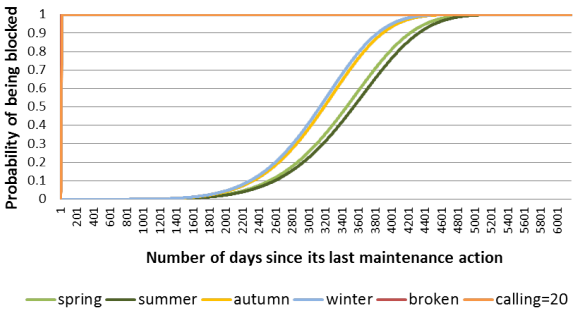
3.2.2. Estimating the process of a gully pot blocking

Ahmad and Kamaruddin (2012) suggest that time-based maintenance is the normal strategy in situations where equipment has a fixed lifespan or predictable failure behaviour. After analysis of historic gully pot records, we model the gully pot blocking process using the Weibull distribution model (Weibull (1951), Ebeling (2004)), from reliability theory. The parameters of this form of Weibull distribution are the shape parameter k, and the scale parameter λ. In our study, we define $k = 6$, which captures a realistically increasing blocking rate over time. The scale parameter λ, capturing lifetime behaviour, is affected by location and seasonal factors, according to a simple linear function:

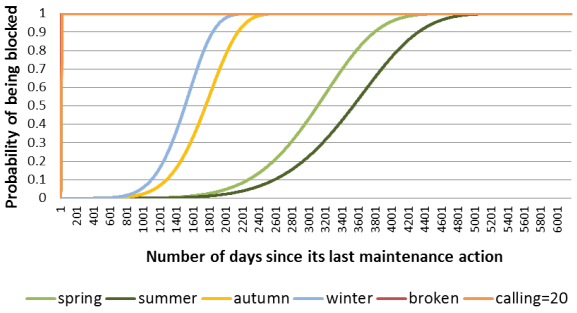
$$\lambda = \begin{cases} 10 & \text{... if gully pot recorded as broken} \\ E_{calling} & \text{... a calling event} \\ max(90, E - \sum_{f \in F} n_f * s_f) & \text{... normal state} \end{cases}$$

$E_{calling}$ represents the expected number of days from a report on a gully pot to its servicing. E is the expected number of days that it would take a normal gully pot to become blocked since its last service. Here, $E = 10.3$ years. F is a set of factors

that may affect gully pot lifetime, such as street type, number of trees nearby, and blown sand effect: n_f represents the effect level from a specific factor $f \in F$ to a gully pot; s_f adjusts the effect from factor f according to seasonal information. For example, if a gully pot is on a street with five deciduous trees nearby, then $n_f = 5$ with $s_f = 93, 1, 389, 433$ in spring, summer, autumn and winter respectively. If a gully pot location is not affected by factor f, we simply assign $n_f = 0$. All values are based on our statistical analysis of the Blackpool data. Fig. 2 illustrates two examples of gully pot lifetime estimation taking account of the surrounding environment.



(a) Example of a gully pot lifetime with 1 tree nearby at different seasons

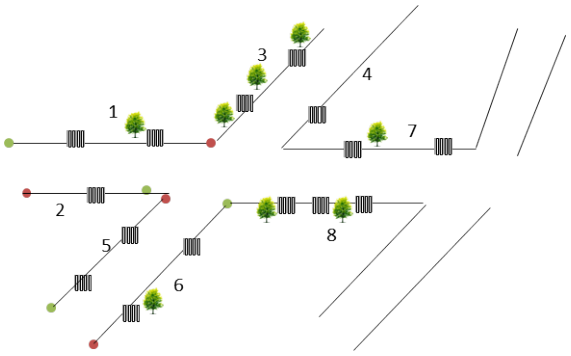


(b) Example of a gully pot lifetime with 5 tree nearby at different seasons

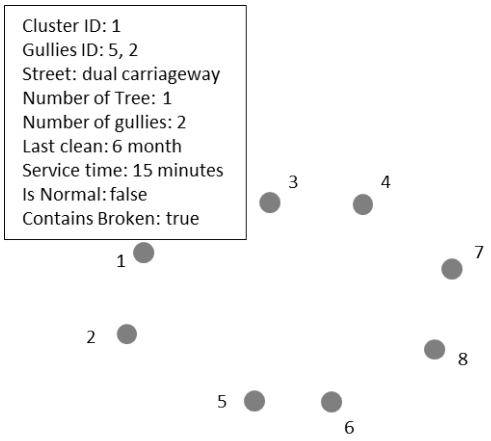
Figure 2: Probability of being blocked since last maintenance action

3.2.3. Reducing the problem size

Blackpool local council would prefer to use an informed maintenance plan at the level of individual gully pots. In recent years, GPS techniques and GIS systems are able to support more precise action tracking and decision making. Our maintenance scheduling problem (28,149 gullies; 36.1 km^2) is significantly larger than existing PVRP case studies. Building geographically tight clusters is the most common step used to reduce a problem (e.g. Cordeau et al. (1997), Blakeley et al. (2003), Tang et al. (2007)).



(a) Example of gully pot on street



(b) Example of grouped information

Figure 3: Reduce problem size

In order to reduce the problem size whilst retaining enough information to build feasible cleaning routes and track gully pot condition, we group gully pots located on the same section of street. As shown in Fig. 3(a), we assume that these gully pots share the same environmental factors. Gully pots in the same group are always scheduled together for preventative maintenance. The service time of a group includes both cleaning time for the gully pots and travelling time inside this section of a road. This representation also maintains traffic distance: for instance, the distance between group point 1 to group point 6, in Fig. 3(b), is the road distance measured from the red node of road 1 to the green

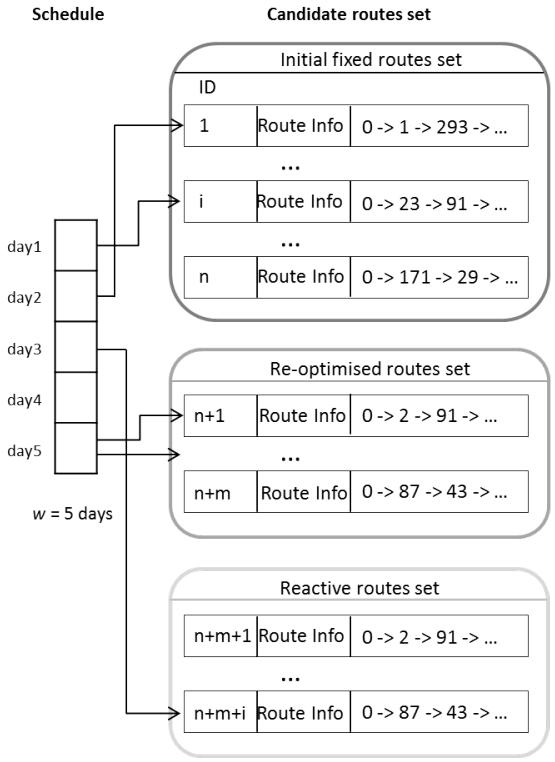


Figure 4: Solution representation and data structure for storing candidate routes

node of road 6, in Fig. 3(a). Furthermore, individual gully pot states (i.e. normal, calling, broken) are still recorded, because unexpected damage or blockage events may happen to any of them: this allows corrective actions to be accurately planned. A gully-pot-cluster is labelled as in normal state only if all the gully pots included are in normal state. The risk of a gully-pot-cluster is the sum of all included gully pots' risk at any given time.

By applying this grouping strategy, we reduce the preventative maintenance problem size from 28,149 to 9,277 points. For corrective actions, routes are built on problematic gully pots, which only compose a small size vehicle routing problem.

3.3. Solution representation

An interesting feature of our problem is that our objective function is designed for a short-term scheduling problem, but the overall aim is to analyse the scheduling impact for long-term risk management. Due to the changing environment and unexpected emerging situations, we cannot assume any repeated schedules between periods. To solve the long-term scheduling problem, a rolling horizon approach is devised, in which the short-term problem is solved repeatedly, given updates to environment and gully pot status.

Fig. 4 shows a data structure used to store the solution. A w days schedule contains selected w IDs of candidate routes. Each route in the candidate set is optimized on distance. A route is composed of an ID, route information and the actual tour. Route information includes up to date gully pot condition which helps to produce schedules:

1. route length;
2. number of gully pots;
3. current route risk, which is the sum of the risk impact for each gully pot multiplied by that pot's current failure rate;
4. tabu tenure l in days, which is used to stop the revisiting of the same preventative route in the near future.

3.4. Candidate routes set management

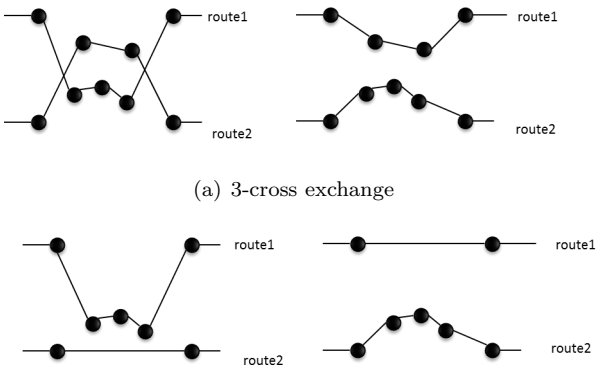
The candidate route set (Fig. 4) consists of an initial fixed routes set, a re-optimized routes set and a reactive routes set. Once the initial fixed routes set is built, these tours will not be changed during execution; it stores initial solutions for preventative maintenance (see Section 3.4.1). Routes in the re-optimized routes set are repeatedly updated during optimisation, as a result of environmental changes that cause gully pot status changes (see Section 3.5). The size of the re-optimized routes set is fixed at m routes. When the set is full and new routes are generated, the oldest route is replaced. Routes in the reactive routes set are built from a set of emerging events and they are all discarded when a schedule solution is executed (see Section 3.4.2).

3.4.1. The initial fixed routes set

Routes optimization is very CPU intensive, especially for such large problems. Constructing routes repeatedly in a rolling planning schema is not efficient. Here, we start by finding a group of optimized candidate routes that can be scheduled directly or adjusted based on updated information before the use of the route in future days. At this stage, we treat the problem as a VRP without considering any risk impact or lifetime information. The objective is to minimize the total travelling distance, with constraints including: 1) all points in the system should be visited exactly once; 2) all

routes should start and end at the deport; 3) no route travelling time should exceed T_{max}.

The vehicle routing solver starts from an initial vehicle routing solution, constructed using the Clarke-Wright (CW) Savings heuristic (Clarke and Wright (1964)). After an initial solution is constructed, the improvement phase uses variable neighbourhood search (Mladenović and Hansen (1997), Hansen et al. (2010)) embedded with i-relocate and i-cross-exchange shaking operators (see Fig.5) and a local search phase. A similar process is used by Hemmelmayr et al. (2009) in their daily VRP solving stage. Here, i-relocate and i-cross-exchange represent that a maximum i number of points in a route are changed in one move. In total, 12 neighbourhoods are implemented. The order of neighbourhoods is i-relocate (i = 1; 2; 3; 4; 5; 6) and then i-cross-exchange (i = 1; 2; 3; 4; 5; 6).



(a) 3-cross exchange

(b) 3-relocate

Figure 5: Inter-routes local moves

In order to enhance the solution quality, a local search strategy is used after a solution is obtained through "shaking". The single route operator, 3-opt (Lin (1965)) is adopted in an iterative first improvement procedure. Only the two modified routes have to be re-optimized.

After finding the optimized VRP solution, we still can not guarantee that every route maximizes the use of the daily time limitation T_{max}. Therefore, for each route in the candidate set S, we try to insert the closest points which are not already included using least cost insertion, until no more points can be inserted without breaking the T_{max} limitation.

3.4.2. The reactive routes set

Before scheduling routes into days, we create candidate routes in the reactive routes set, based on emerging events information. During the last w days, a normal callings and b broken reports are received. Calling reports that have not been addressed and a new calling reports make up the set V_{calls}. In the same way, we also get a set V_{broken}. When a call is received, we register the cluster ID (see Section 3.2.3) so that the schedule can inspect gully pots around the reportedly problematic ones; however, when broken pots are discovered through preventative maintenance or inspection, we register them individually.

The VRP solver described in Section 3.4.1 is used for both V_{calls} and V_{broken} to create candidate route sets S_{calls} and S_{broken}, respectively. Each route in S_{calls} is treated as an opportunity to clean more normal-state gully pots on the journey, as the same vehicle is used for the task. So, for each route in S_{calls}, we try to insert the closest cluster-points that are in a normal state, and whose time since last service is longer than 30 days. We use least cost insertion, until no more points can be inserted without breaking the schedule duration constraint based on T_{max}.

At this point, we have a candidate routes set (including preventative routes, routes that mostly contain reported gullies and routes that only contain broken gullies) optimized in distance:

$$S_{all} = S_{fixed} \cup S_{calls} \cup S_{broken}$$

3.5. Produce schedule

To produce a maintenance schedule in continuous time, Fig. 6 illustrates an overview of system information flow and Algorithm 3.1 describes a rolling horizon optimiser that automatically selects appropriate maintenance actions (either preventative or corrective) for the upcoming period.

3.5.1. Initialization

The initial schedule simply chooses the w number of routes with tabu tenure l_s equals zero, from all candidate routes, S_{all}, with highest risk, $\sum_{i \in s} r_i P_i(today)$.

3.5.2. Improve the schedule using BEBO heuristic

The improvement stage is developed on a tabu search based hyperheuristic method – binary exponential back off (BEBO), proposed by Remde et al.

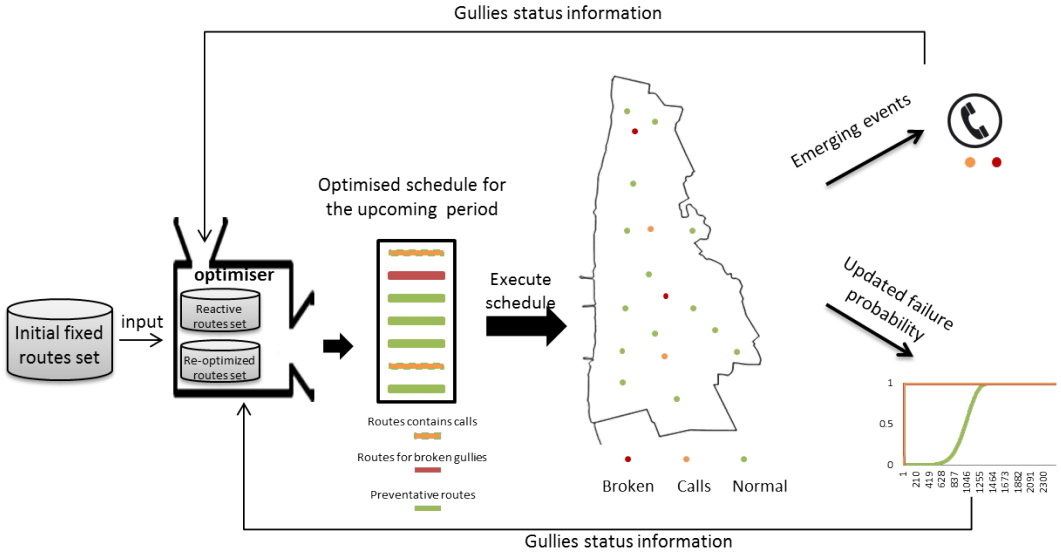


Figure 6: Overview of system operation

Algorithm 3.1 Rolling horizon optimiser – algorithm sketch

Define:

S_{fixed} is the initial fixed route set containing distance optimized routes.

S_{reopt} is a set of distance optimized routes that are updated during the search according to the recent gully pots risk information; initially $S_{reopt} = \emptyset$

l_s is a tabu tenure of route s in days, to stop revisiting of this route in near future (Section 3.3).

u_s is a flag parameter to prevent cyclic testing of route s when using the scheduling-related LLHs (i.e. LLH_3,LLH_4 in Section 3.5.2)

Rolling horizon repeat every w **days**:

1. Generate S_{calls} and S_{broken} based on emerging events. $\forall s \in S_{calls} \cup S_{broken}, l_s = 0$
2. Get the candidate routes set $S_{all} = S_{fixed} \cup S_{reopt} \cup S_{calls} \cup S_{broken}$. $\forall s \in S_{all}, u_s = false$
3. Generate w days schedule solution x that minimizes the objective function 1 (Sect. 3.5.1 – 3.5.3).
4. Update routes $s \in S_{fixed} \cup S_{reopt}$ risk information based on condition changing of gullies;
5. If any route $s \in S_{fixed} \cup S_{reopt}$ is scheduled in x, $l_s = 30(days)$, otherwise $l_s = l_s - w$

(2009). BEBO has the fundamental structure of hyperheuristic search strategy – a trial set of low level heuristics (LLHs) and systematic rules that control the usage of each LLH. BEBO uses dynamically adapted tabu tenures (Glover and Laguna (2013)) during the search process, especially useful when large neighbourhoods are involved. If a LLH performs poorly in a recent search, it is disabled for a number of iterations. If the LLH performs poorly continuously, the number of forbidden iterations increases. The detailed searching framework is shown in Algorithm 3.2 (Remde et al. (2012)).

Originally, hyperheuristics were designed for the purpose of automatically choosing the right low level search strategy/strategies at each decision point (Cowling et al. (2001)). There is good evidence that hyperheuristics can be successfully applied to various combinatorial problems, such as timetabling (Burke et al. (2010a), Bai et al. (2012)) and vehicle routing (Garrido and Riff (2010), Misir (2011), Walker et al. (2012)). Summaries of state of the art hyperheuristic techniques can be found in survey papers by Özcan et al. (2008), Chakhlevitch and Cowling (2008), Burke et al. (2010b, 2013).

Apart from the hyperheuristic framework, a well-designed set of LLHs is crucial to successfully applying a hyperheuristic. In our implementation, the LLHs are designed at two levels: 1) route-related moves that modify routes by changing segments or points in or between routes; 2)

schedule-related moves that assign an optimized route to a day. The value of a solution is measured by the objective function in Equation 1, above.

▼ Route related moves:

The following route related moves are only applied to preventative routes and routes that contain mostly reported calls. Fixing broken gully pots is carried out by a different vehicle. These reactive routes $s \in S_{broken}$ are constructed as described in Section 3.4.2 and no more route structure optimization is processed.

LLH_1. i-cross exchange. For any two scheduled routes r_1 and r_2, apply i-cross exchange. If any resulting route visits one point more than once, the points adjacent to longer edges are removed. Moves are examined for each pair of routes in a nested loop, the first yielding an improvement being implemented. $(1 \leq i \leq 5)$.

LLH_2. i-worst point insertion $(5 \leq i \leq 20)$. This LLH improves the next w days' scheduled routes by finding the i highest risk points not appearing in the current schedule solution x. These i points are then inserted into the w days schedule using a cheapest insertion heuristic with a relaxed time limit. If any target route in w now exceeds the T_{max} limitation, we repeatedly remove the best-condition point from that route until it becomes feasible.

The two LLHs above keep a copy of the original routes and generate new routes through operations. New routes are stored in the re-optimized routes set S_{reopt}. Though these modified routes may not generate improvements for the current iteration or the current short planning horizon, they normally contain relatively high risk cluster-points in recent time. Hence, they are still likely to be picked up using schedule related moves later or contribute to the near-future plan.

▼ Schedule related moves:

LLH_3. n-replace schedule $(1 \leq n \leq w)$. Replace the last n days' schedule with n other routes from the candidate set S_{all}, that are not included in the current solution, and whose tabu tenures l_s equals zero and has not been tested during the search $(u_s = false)$. We sort the candidate set S_{all} to check the higher risk routes first as these moves are more likely to produce improvements. Algorithm 3.3 presents the pseudo code of LLH_3.

Algorithm 3.2 Bebo hyperheuristic

Define:
x is the current solution;
LLH_i is a low level heuristic;
$\triangle(x, LLH_i)$ returns the new value of the objective function 1 from applying LLH_i to current solution x;
$tabu_i$ is the tabu tenure of LLH_i
$backoff_{min} = 5$ is the minimum backoff value
$backoff_i$ is the backoff value of LLH_i, where $backoff_i \geq backoff_{min}$
for all i **do**
  set $backoff_i = backoff_{min}$
  $tabu_i = 0$ *# in our implementation, we allow all LLHs to try at least once at the beginning*
end for
while $\exists i$ that $tabu_i = 0$ **do**
  $bestvalue = x.value$
  for all LLH_i **do**
    if $tabu_i = 0$ **then**
      if $\triangle(x, LLH_i) < x.value$ **then**
        $backoff_i = backoff_{min}$
        if $\triangle(x, LLH_i) < bestvalue$ **then**
          $bestvalue = \triangle(x, LLH_i)$
          $besti = i$
        end if
      else
        $backoff_i = backoff_i * 2$
        choose $tabu_i$ randomly from $\{0, 1, ..., backoff_i\}$
      end if
    else
      $tabu_i = tabu_i - 1$
    end if
  end for
  if $bestvalue < x.value$ **then**
    $x \Leftarrow apply(x, LLH_{besti})$
  end if
end while

Algorithm 3.3 n-replace heuristic

```
S_all is a list of candidates routes sorted by risk in descending order
for each day i in the last n days' schedule do
  for each route s in S_all where u_s = false && l_s = 0 && s ∉ x do
    x' = replace the route scheduled in day i with s;
    if x'.value < x.value then
      x = x'
      u_s = true
      break
    end if
  end for
end for
return x
```

LLH_4. n-replace schedule random ($1 \leq n \leq w-1$). Same as LLH_3, except that we choose the n day's schedule to replace randomly, instead of the last n day's schedule.

LLH_5. switch two days' schedule (see Algorithm 3.4). First improvement scheme is applied.

Algorithm 3.4 switch heuristic

```
for i = 0; i < x.length; i = i + 1 do
  for j = i + 1; j < x.length; j = j + 1 do
    x' = switch the ith and jth days' schedule;
    if x'.value < x.value then
      return x'
    end if
  end for
end for
return x
```

LLH_6. Pop up (Algorithm 3.5). Pop up ith day's schedule to a target position j. For example, one neighbour of solution 1,2,3,4 can be 1,4,2,3 by popping up 4 to the second position. In Algorithm 3.5, entry=w and stop=1 are used in following experiments.

In summary, if we need to produce a $w = 7$ days schedule, in total 36 LLHs will be called. The LLHs set contains both route structure adaptation and schedule modification according to risk estimation. Our preliminary experiments show that all the LLHs contribute to the final solution quality. Among them, LLH_2 makes the most improvements. Also, LLH_2 helps the solver continuously add new

Algorithm 3.5 Pop up (entry, stop)

```
Define: entry: the route to pop up
        stop: the target day;
for i = entry; i > stop; i = i − 1 do
  for j = i − 1; j ≥ stop; j = j − 1 do
    x' = pop up ith day's schedule to jth day;
    if x'.value < x.value then
      return x'
    end if
  end for
end for
return x
```

elements to the candidate routes set. Our LLHs do not allow any individual route to visit one point more than once. However there is no rule to eliminate a solution that contains a cluster-point more than once during the period w: our experiments suggest that such a sub-optimal solution is easy for the algorithm to improve using LLH_2, LLH_3, LLH_4, which is thus rarely seen in practice. If a resulting solution suggests to visit a point more than once within w days, the heuristic is opportunistically visiting a recently cleaned gully that lies close to the current route.

3.5.3. Improve current solution by partial rebuilding

Given a current solution x returned from the BEBO improvement stage, we reinitialize x by partly destroying and rebuilding x. Then the BEBO improvement and reinitialization repeats for a given CPU time. The global best solution is remembered.

Destroy: For a w days schedule solution, we randomly remove y days of the schedule, where $y \leq w/2$;

Rebuild: Here, we build y new routes that can replace the y removed schedules. First, from the optimized routes information stored in the fixed memory, we know the average number of points $\bar{n}$ included in a route. We then select n_{worst} number of points with the highest risk under current environment and n_{random} random points that have not been visited in the $w - y$ unchanged scheduled routes, where $n_{random} = n_{worst} = \bar{n} * y/2$. Next, the entire process in Section 3.4.1 is applied to the selected points, resulting in z distance-optimized routes, which are stored in the the re-optimized routes set S_{reopt}. Finally, y out of the z routes are

randomly assigned to replace the removed schedules.

4. Experiment

In this section, we summarise the background of our problem and simulation. Then, we determine the rolling planning horizon by experimenting with its effect on risk management under different environmental conditions. Finally, we consider how different maintenance policies affect the surface water flooding risk due to blocked gully pots in the long term. All simulations were implemented in C# and executed on a cluster composed of 8 Windows computers with 8 cores, Intel Xeon E3-1230 CPU and 16GB RAM.

4.1. Data & Parameters

4.1.1. Simulation settings

Gully pot information comprises location, surrounding properties, nearby trees and historical maintenance actions from Blackpool local council, a client of Gaist Solutions Ltd.

1. Total number of gully pots in the system: 28,149.
2. Broken events: according to the records, on average about 1.8% of gully pots are broken every year. In our simulation, this is represented by each gully pot becoming broken randomly with probability 0.00005 per day.
3. Accessibility: on average, the records show that about 8.3% of gully pots are not serviced during maintenance each year. In a normal cleaning day, the maintenance team is able to visit about 80 gully pots. In the simulation, we assume that the probability of a gully pot being inaccessible during preventative maintenance is 0.068. For corrective actions, including servicing both calls and broken gully pots, we assume the team always has access to the gully pot.
4. Blocking probability: a gully pot lifetime is estimated by a Weibull distribution described in Section 3.2.2. Every day, each gully pot has a probability of becoming blocked according to its failure rate function $h_i(d) = \frac{R_i(d-1)-R_i(d)}{R_i(d-1)}$, where $R_i(d) = 1 - F_i(d)$ is the reliability function.
5. Seasonal factors F: the Blackpool data only allows us to include trees and leaf-fall in our simulation. Seasonal factors related to the number of trees nearby highly affect the lifetime of gully pots, and on average, each gully pot is affected by 0.4 trees in Blackpool.
6. Calls: about 1700 calls are received every year by the Blackpool gully maintenance team, and most of the calls concern blocked or damaged gully pots. Over 50% of all calls occur during the autumn, as shown in Figure 7. Our statistical analysis determined that, to match the resident calling behaviour in our simulation, the probability of receiving a call if a gully pot is already broken or blocked is $p_{calls}(i) = \{0.0033, 0.005, 0.0056, 0.002\}$ for spring through winter, respectively. If a gully pot is not broken, there is still a small chance that a call is received, related to its current condition. The simulation probability is $p_{calls}(i) = P_i(d) * \gamma$, where $\gamma = 10.62$ has been measured experimentally to adjust the calling probability to match the real data.
7. Other issues: as well as broken gullies reported by residents, damage is also found during preventative maintenance. In this case, the simulation registers the broken gully and schedules it on a later day.

These parameters have been discussed with Gaist Solutions Ltd. and agreed to be a realistic representation of gully-pot behaviour in Blackpool.

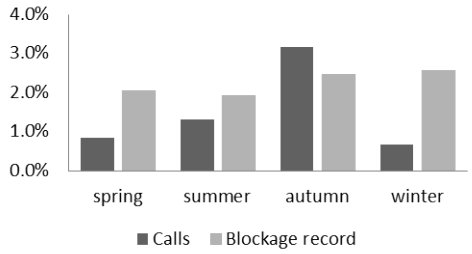


Figure 7: Percentage of calling evens and blockages recorded in different seasons, for the gully-pot system in Blackpool

4.1.2. Search parameters setting

The BEBO heuristic described in Section 3.5.2 is parameter-free, since all LLHs are given and it always chooses the best LLH for each decision point.

The termination criterion of the entire search process composed by BEBO and reinitialisation is

controlled by a pre-set CPU time. Many heuristic search strategies find good solutions in the very early stages, but to find more improvements becomes harder and harder. To avoid either too early termination or unnecessary CPU consumption, we test the effects of limited computation time for various sizes of planning horizon, w. According to our experiments, about 0.002, 15, 68, 319 and 1189 minutes are required respectively for planning horizons $w = \{1, 5, 7, 10, 14\}$ to achieve results that are within 2% of the best found solutions in preliminary experiments run over 48 hours of CPU times (see Figure 8). These CPU time limitations are used in the subsequent experiments.

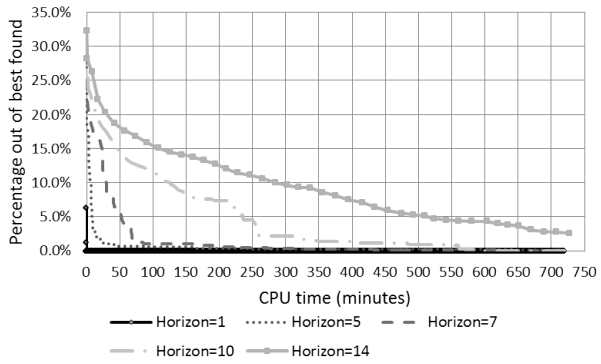


Figure 8: Effects of limiting computation time for different planning horizons

In the long term rolling planning process, we save the newest generated routes in a re-optimized routes set S_{reopt} of size m. Large values of m result in a more diverse set of routes, which may lead to a better solution. However, if m is too large, the increased CPU time (for schedule-related LLHs (Section 3.5.2) to find their local optima) does not yield better solutions in the time available. The route diversity due to larger m contains too many old updates during the search, which increases the searching complexity. If m is too small, route-related moves repeatedly generate the same or very similar routes. For our case, $m = 25\%$ of the initial fixed routes set size (see Figure 4) is found to give the best balance between these two effects in preliminary experiments.

4.2. Impact of planing horizon w on risk management in different environments

Gully pot lifetimes are affected by seasonal factors. In addition, peoples' reporting behaviour is different at different times of year. A short planning horizon may result in many reactive actions, and require more frequent information updates, whereas a longer planning horizon is better at balancing preventative and corrective maintenance. However, when w is too large, it leads to a plan based on insufficiently up-to-date information.

This section explores the impact of the planning horizon w on the maintenance performance in four seasons with the gully-pot system in either stable or recovering states. As shown in Table 2, a stable state assumes that the entire system is well maintained and the number of days since last maintenance action of each gully is uniformly distributed across 1.5 years. The recovering state assumes that the system has had bad maintenance and the number of days since the last maintenance action for each gully is uniformly distributed within 3 years. For each scenario, $w = \{1, 5, 7, 10\}$ is tested.

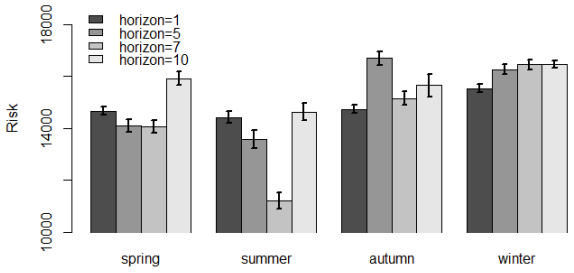
Figure 9 shows the average daily surface water flooding risk caused by clogged gullies in Blackpool by using different planning horizons under different scenarios. We can be relatively sure that there is a genuine difference in risk, when both the mean values and the 95% confidence intervals differ. In the stable scenarios (Fig. 9(a)), $w = 5$ and $w = 7$ perform better than other settings during spring and summer. Over autumn and winter, when the number of blocked gully pots and calls significantly increases, $w = 1$ produces the best schedules, as it updates system and environment information most frequently. $w = 10$ performs badly in all seasons, due to lack of up-to-date information.

In the recovery scenarios (Fig. 9(b)), the overall risk is about 2 to 3 times greater than when the system is in the corresponding stable state. In particular, if there is a lack of maintenance in autumn, this may lead to serious consequences. Again, $w = 1$ always produces the best schedules in the recovery state. This is because there is a significant number of emerging situations every day. Updating the system and environment information every day brings considerable advantages. In the recovery scenario, it is difficult to identify a single best value among $w = \{5, 7, 10\}$; it is hard to balance the preventative and reactive actions by adjusting planning horizons in relatively dynamic situations.

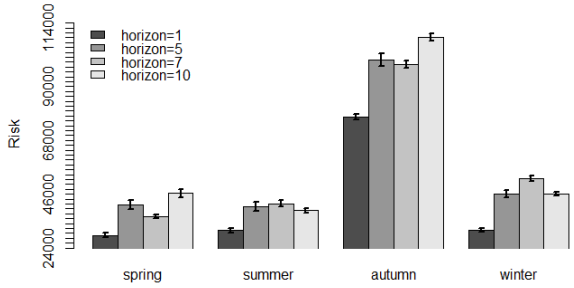
Table 3 presents scheduling performance in terms of corrective actions. As we described in Section 3, our solution does not impose hard constraints on the time taken to respond to residents' calls. Instead, the hyperheuristics automatically choose

Table 2: Environment settings

State	Definition	Years since last maintenance
Stable	Based on the real-world situation, running simulation of maintenance actions for a long period until the overall system risk becomes stable; used as the initial state for all stable scenarios	0 to 1.5
Recovery	Start with very poor gully-pot conditions, and entire system in a high risk state	0 to 3



(a) Stable



(b) Recovering

Figure 9: Impact from planing horizon to maintenance performance in different environments. Error bars show 95% confidence interval on each mean.

maintenance actions that minimize the entire system's risk. On average, in a stable state all tested planning horizons react to emerging events in less than 7 days. In the recovery state, $w = 1$ gives the fastest reaction to these emerging problem gully pots. However, even with $w = 1$ there are big challenges in the autumn period, when the average delay between identification and correction of a problematic gully pot is 34 days.

4.3. Effect of maintenance policies on risk in continuous time

Our essential aim is to reduce the surface water flooding risk for the entire city in continuous time. In the previous section, we seek the best-performing rolling planning horizon. $w = 1$ requires the shortest computation time and produces the best schedule when the system is under pressure, but collecting the system and environment information every day is not feasible in real-world team management. When the gully-pot system is in its stable state, $w = 7$ shows the best ability to cope with seasonal changes. After consultation with Gaist Solutions Ltd, $w = 7$ is applied in the long period maintenance policy testing, since this balances team management requirements and scheduling performance.

In order to test the impact of how we manage preventative and corrective maintenance, we designed six policies that combine preventative and corrective actions with different rules. In these experiments, all scheduled routes are optimized on distance.

- Policy0: Pure reactive policy. Every week, we produce a $w = 7$ days schedule for reported problematic gully pots only, according to up-to-date information. Priority is given to the emerging events with highest risk. After finishing these planned tasks, we take a rest until the plan for the next week is produced.

Table 3: The effect of planning horizon on corrective maintenance performance. Emergings per day: the average number of identified problematic gully pots.

	Stable							
	Spring		**Summer**		**Autumn**		**Winter**	
	Average response	Emergings per day	Average response	Emergings per day	Average response	Emergings per day	Average response	Emergings per day
1	1.56	0.53	1.68	0.64	2.55	5.18	3.00	2.54
5	3.82	0.67	3.97	0.33	3.88	4.85	4.24	2.37
7	4.58	0.57	4.28	0.51	4.36	4.87	4.97	2.51
10	5.98	0.63	6.23	0.43	4.58	4.80	3.71	2.32
	Recover							
1	14.74	25.02	14.22	24.27	34.41	65.22	16.59	28.29
5	18.26	24.73	18.82	23.22	36.80	64.52	25.04	29.33
7	20.07	23.41	22.46	23.01	36.40	65.06	22.15	28.38
10	17.63	23.26	19.86	23.07	36.12	63.44	19.32	27.69

- Policy01: Pure reactive policy. Every day, we produce a $w = 7$ days schedule for reported problematic gully pots only, according to up-to-date information. Only the first day schedule is executed, then we replan for the following week.
- Policy1: Pure reactive policy 0 in autumn, predictive schedule (see Section 3) with planning horizon $w = 7$ in other seasons.
- Policy2: Predictive schedule, introduced in Section 3, for all seasons.
- Policy3: Fixed manual schedule. All preventative routes are generated at the beginning of a year, giving the routes stored in fixed memory, see Fig. 4. These routes are arranged in descending order of risk measured at the initial time. Every week, we use the first two days to deal with emerging events and use the remaining 5 days to deploy preventative actions, in order. During corrective time, we give priority to routes with the highest risk, $\sum_{i \in s} r_i P_i(today)$.
- Policy4: Dynamic manual schedule. Similar to policy 3, but priority is given to corrective maintenance. We firstly serve all known problematic gully pots. If there are still some days left in this week, we carry out the pre-ordered preventative maintenance routes. This policy is widely used in real-world gully pot maintenance programmes.

We evaluate the performance of each policy from three aspects: overall risk management, agility to emerging events, and running cost. These six policies are firstly tested in a stable state and then we test their recovery speeds in a variety of bad initial situations. The daily risk is evaluated from the actual blocked and broken gully pots with their associated risk impact.

4.3.1. Performance in the stable state

We run each policy on the Blackpool gully-pot system over four years, with corresponding seasonal settings and residents' reporting behaviour. Five random runs are carried out for each policy. We evaluate the average daily risk based on these experiments.

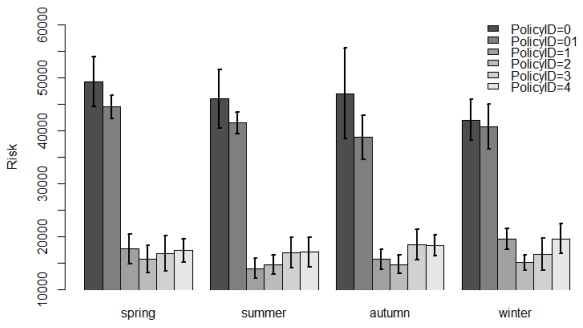


Figure 10: Policy performance in stable state. Error bars show one standard deviation of daily risks for each season.

Risk management. Figure 10 shows the average daily risk when applying the different maintenance policies in a stable state. Pure reactive policy 0 produces the highest risk all the time, and is about

three times worse than any preventative and corrective combined policy. Even if we reschedule every day (policy 01), pure reactive maintenance still preforms significantly worse than other policies. The performance of pure reactive policies in autumn is not significantly worse than their performance in other seasons. However, their data shows very big deviations in autumn, which suggests large fluctuations happen. In the daily performance tracking, we find serious risk increases at the beginning of autumn, due to the lack of maintenance in other seasons and environmental factors. Also, in autumn, residents' reporting behaviour helps to prompt a large number of reactive actions.

From policy 1 to 4, the predictive scheduling strategy (policy 2) achieves the best overall performance. It is significantly better than manual scheduling in summer, autumn and winter. In spring, there is not much difference in applying any of the preventative plus corrective policies. To track the daily risk change over time, we apply these four policies in exactly the same environment simulation for four years. As illustrated in Figure 11, policy 4 is used as a base line and the other three policies are compared against it. When applying policy 4 in a stable state, the estimated cost due to surface flooding risk is £18,082 on average per day. By just rearranging the preventative and corrective tasks, policy 3 achieves an average risk decrease of about 12% per day, but always giving priority to emerging events may lead to poor working efficiency. The best result is for policy 2, which produces schedules that out perform the base line (policy 4) in 91% of days over 4 years; on average, policy 2 decreases risk by about 17% per day.

Agility. Table 4 presents the average number of days to respond to calls. All policies except policy 3 are able to react to emerging calls in less than 5 days on average. Policy 3 uses a very straightforward scheduling rule, which may be good for team management, but shows serious latency for emerging requests. When only applying reactive actions (policy 0 and 01), on average about 3 times more residents' calls are received per day. This also exposes one reason for the poor performance of these policies in risk management (Fig. 10): lack of preventative maintenance leads to more corrective maintenance.

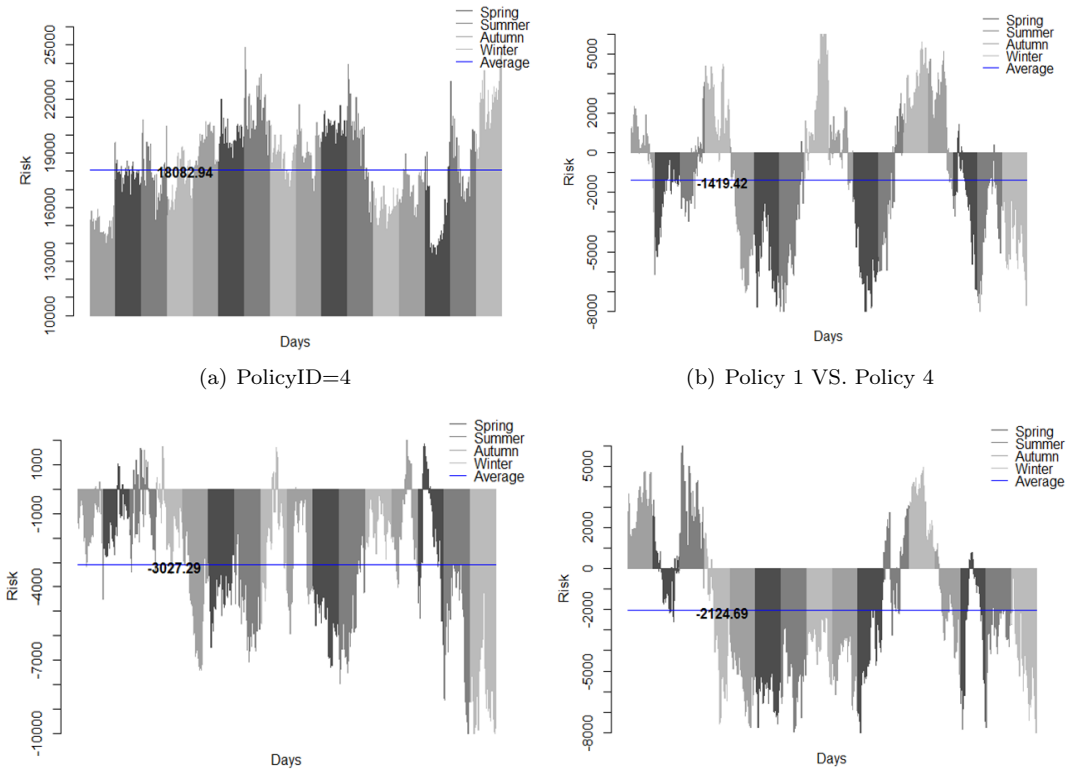
Working efficiency analysis. To discover how the predictive schedule strategy (policy 2) out performs other policies, we focus on time usage and work efficiency. Figure 12 illustrates the percentage of time spent in different types of activity. First, we can see that the reactive policy 0 shows high dependence on resident reports, resulting in working time of only about 45% percent during spring, summer and winter. Policy 3 follows a very straightforward rule, to do maintenance throughout the year. The fixed rule lacks the ability to adapt to seasonal changes. As Figure 10 shows, policy 3 has the largest fluctuations in all seasons compared to other preventative and corrective combined policies.

Table 4: Agility analysis of different maintenance policies

	Average response	**Emergings per day**
Policy 0	4.88	11.14
Policy 01	2.24	10.88
Policy 1	3.93	4.31
Policy 2	4.34	3.30
Policy 3	**20.82**	2.83
Policy 4	4.67	3.19

The time usage distributions of policy 2 and the manual schedule policy 4 show very similar patterns in Figure 12. Table 5 compares the daily working efficiency of policy 2 and 4. On average, policy 2 manages to service 10 more gully pots every day within the same working time constraints. One reason is because policy 2 treats the resident calls and normal preventative maintenance together, so more efficient routes can be found and emerging blockages can be solved at the same time. Comparing to the fixed preventative routes managed by policy 4, policy 2 always attempts to insert more high risk gully pots into the current scheduled routes (Section 3.5.2 $LLH2$), which results in automatically rescheduling of any missed gullies from previous preventative maintenance. Figure 13 illustrates further evidence that policy 2 produces better schedules. Comparing Figure 1 (the gully pot risk impact map of Blackpool) and Figure 13 (the service frequency map under policies 2 & 4), we find policy 2 successfully targets the geographical areas which have been evaluated as highest risk. In contrast, policy 4 schedules the service times more evenly, which results in too many visits to low criticality areas.

Cost. using Blackpool's current operational costs (Table 6), we can estimate the annual cost of each maintenance policy. This allows us to explore the



(a) PolicyID=4

(b) Policy 1 VS. Policy 4

(c) Policy 2 VS. Policy 4

(d) Policy 3 VS. Policy 4

Figure 11: Daily risk change over 4 years in a stable state using 4 types of preventative plus corrective maintenance policy.

Table 5: Average number of gully pots serviced per day by policy 2 and 4

	Spring	**Summer**	**Autumn**	**Winter**
Policy2	81.90	82.09	83.41	81.42
Policy4	71.03	71.45	74.60	72.28

cost of extra effort required for preventative maintenance.

Table 6: Operation costs of gully-pot maintenance

	Cost	**Unit**
Travelling	£0.28	per km
Vehicle maintenance	£20,000	per year
Human resource	£56,000	per year
Preventative	£3.25	per gully
Calls response	£19.00	per gully
Broken	£225.00	per gully

The cost estimates for the different policies is shown in Figure 14. All of the preventative and corrective combined policies show expenditure of £280,000 to £300,000 annually. This means that, compared to the pure reactive policy 0, an extra 10% of expenditure could reduce potential risk by as much as a factor of 3, over time (Figure 10).

Comparing the predictive policy 2 to the current manual policy 4, about £8,000 more would need to be spent each year, due to the additional preventative work. However, these extra preventative actions would result in about £3,000 of risk reduction every day, or over £1 million per year.

Due to data limitations, our current simulation of gully pot breakage behaviour uses a fixed probability, giving roughly 500 broken gully pots a year, generated at random times. This simplistic breakage regime, which is the same under each policy, results in all policies presenting similar effort to tackle broken gully pots. In practice, policies with regular preventative maintenance would slow deterioration, and might decrease the chance of breakage. We would expect a more realistic model of breakage probability to reduce the apparent cost of policies 1 to 4 to less than the cost of policies 0 and 01.

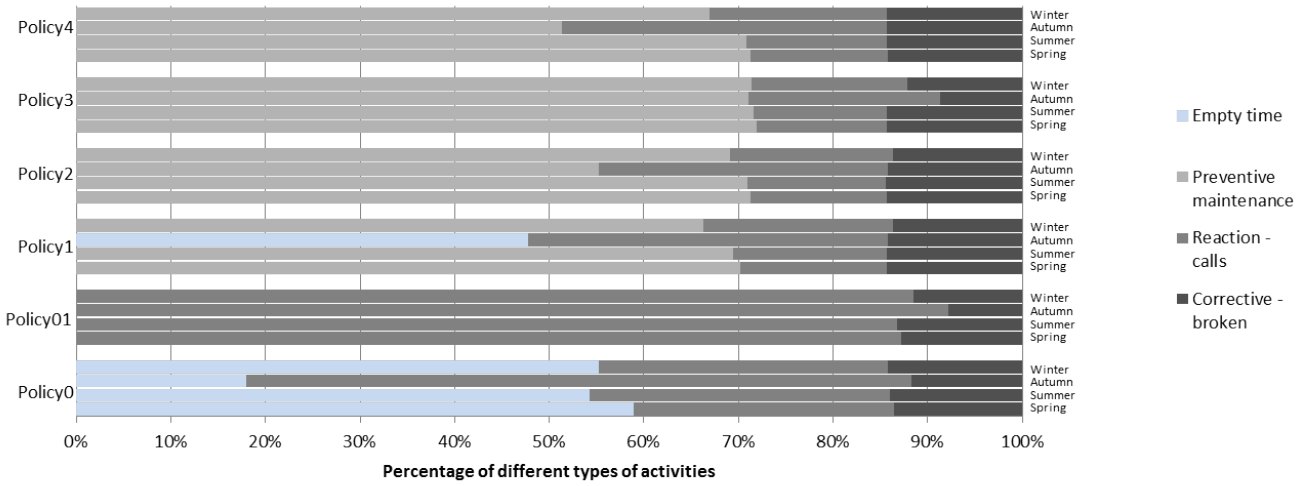


Figure 12: How different policies use their time to do maintenance

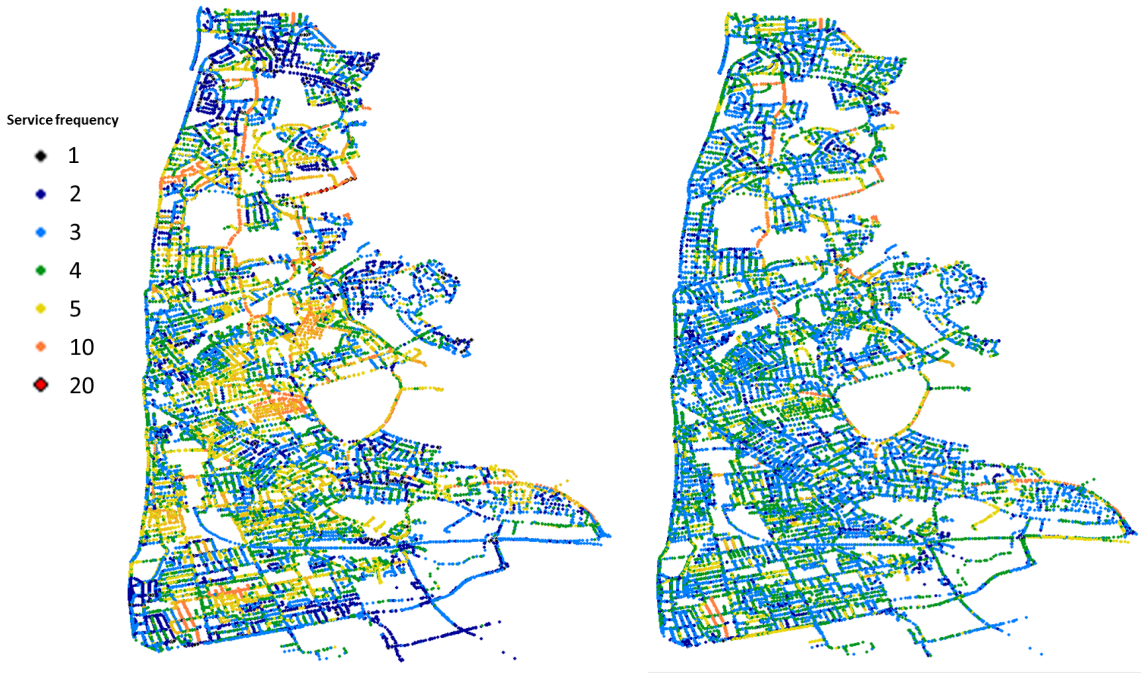
In conclusion, preventative maintenance could significantly ameliorate the surface water flooding risk caused by blocked gully pots at reasonable additional cost; these costs are more than justified by service quality improvement.

4.3.2. Performance in recovery state

Here, we test the robustness of each policy by starting from a very bad initial condition. We explore how long it takes for each maintenance policy to take the system from a poor initial state to a stable state. The average risk of applying each policy in a stable state (Section 4.3.1) is used as the policy's base line. As presented in Table 7, four scenarios are tested. We use two parameters, "since last maintenance action θ" and "percentage of broken gully pots" to control the system's initial state. For each scenario settings, we report the average of 10 runs of a two-year simulation.

Recovery speed. From Figure 15, we can see that the initial situation in scenario 2 is very close to the stable state of reactive policy 0. Comparing scenario 1 to scenario 2, the overall shortage of preventative maintenance is more difficult to recover from than a small amount of very broken gully pots in the system. On average, policies 1 to 4 need about 7 months to restore the system to its stable state in scenario 2 (see Fig. 15(b)), whilst they need about 19 months to recover from initial situation in scenario 1 (see Fig. 15(a)). Comparing policies 1 to 4 in scenario 1 and 3, we can see that policies 1 and 2 perform better than both of the manual policies 3 and 4 in terms of percentage risk increase. The robustness of both manual policies is considerably worse than that of policies 1 and 2, especially during the autumn period in the first year. Comparing the performance of policies 3 and 4, the fixed schedule strategy (policy 3) is a lot worse than the more flexible strategy (policy 4).

Activity changes during recovery stage. To recover from different situations, policies 1 to 4 utilize their time in different ways. Figure 16 presents the time usage of each policy during the first year of the recovery stage. Policy 3 has fixed amount of preventative time, about 71%, through all scenarios. However, it still adjusts the remaining 29% of corrective action time to face different types of emerging events (including calls and broken gully pots). Comparing policies 3 and 4 in scenario 1, 3 and 4, the relatively more flexible policy 4 almost stops its preventative actions except in winter. This flexibility helps policy 4 to recover the system faster in the early stage and results in less total damage during the recovery stage. Interestingly, both policies 3 and 4 take similar amounts of total time to recover the entire system in all scenarios: the rate of recovery for policy 4 slows over time. The predictive policy 2 balances its preventative and corrective time, and is between policies 3 and 4. The balanced strategy results in a steady recovery process; even though it only does corrective work during autumn period (like policy 1) and has some resting



(a) Policy 2 (b) Policy 4

Figure 13: Geographic distribution of service frequency in 4 years simulation

Table 7: Initial conditions for different recovery scenarios. Two parameters, "since last maintenance action" and "percentage of broken gully pots" set the system's initial state: for all gully pots, the days since their last service are evenly distributed in θ years. We randomly assign x percent of gully pots to be in the broken state

	Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**
Since last maintenance θ	3 years	1.5 years	3 years	4 years
Initial broken gully pots	0.7%	2%	2%	3%

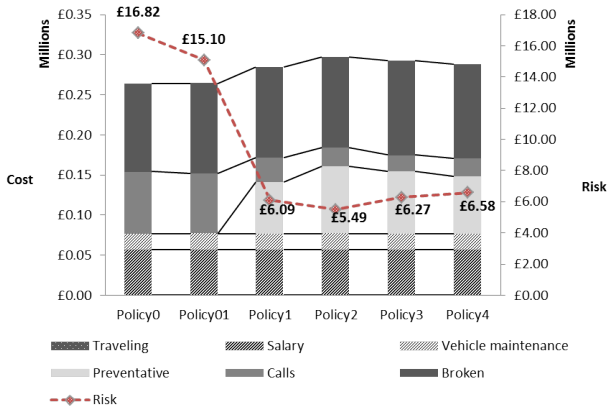


Figure 14: Annual operation cost and surface water flooding risk caused by clogged gully pots for the different policies

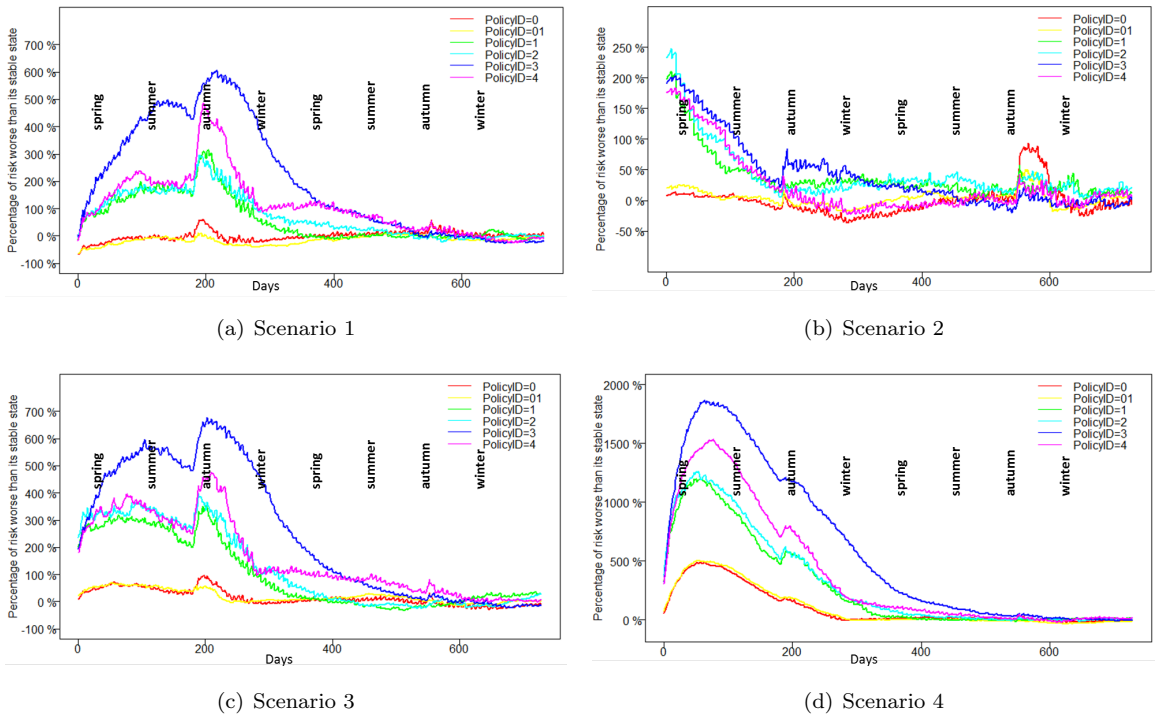
days, the overall performance is not affected.

4.4. What-if questions

All the experiments introduced in the previous sections are based on the real-world scenarios. In this section, we test three hypotheses, which uncover potential weaknesses of our scheduling approach, and suggest future investment directions to improve maintenance performance. Experiments use policy 2 plus the current manual approach, policy 4.

4.4.1. What if we do not have information on risk impact?

Section 3.2.1 describes the method to collect and estimate each gully pot's risk impact, which estimates the consequences of surface water flooding due to clogged gully pots. However, not every local



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Figure 15: Recovery speed using different policies. The percentage of risk is calculated as $(r - \widetilde{r})/\widetilde{r} * 100\%$, where r represents the daily surface flooding risk and $\widetilde{r}$ is the average daily risk of applying the corresponding policy in its stable state (see Section 4.3.1).

council records the information. Figure 17 compares the performance of policies 2 and 4 with the same policies operating when there is no risk impact information, labelled policy 2* and policy 4*. Policy 2* results in a much higher risk than policy 2 (Figure 13(a)); in fact, the lack of risk impact data means that policy 2 becomes similar to policy 4: all gully pots are serviced evenly. For the same reason, missing risk impact information (policy 4*) has a minimal effect on performance compared to policy 4.

4.4.2. What if all gullies are accessible – the impact of parking issues?

In Section 4.1.1, the simulation has about 8.3% of gully pots inaccessible, modelling the effect of parking. This decreases the maintenance working efficiency. Potential strategies can be proposed such as banning parking when a maintenance visit is scheduled, but this increases management complexity and residents' complaints. Here, we explore the impact that inaccessible gully pots has, by comparing results for policies 2 and 4 with results for simulations with zero inaccessible gully pots (policy2** and policy4**). From figure 17, for the current manual schedule strategy (policy 4), there is about a 13% risk decrease if all gully pots are accessible. However, there is no difference between policy 2 and policy 2**. Policy 2 is more able to cope with the current parking issues, because it flexibly reschedules preventative maintenance of inaccessible gully pots.

4.4.3. What if we could do condition-based maintenance (CBM)?

Improving low-cost sensor techniques make it potentially feasible to continuously monitor gully-pot condition. This would allow our schedule strategies to be combined with CBM, discussed in Section 2.1. Currently, we only find out that a gully pot is blocked or broken if it is found during preventative maintenance or reported; because of this incomplete system information, it is difficult to produce truly optimal schedules. To demonstrate the importance of timely blockage or breakage information, we simulate a scenario in which all problematic gully pots are known immediately. From the results in Figure 17, both of the policies achieve dramatic

		Scenario1				Scenario2				Scenario3				Scenario4			
		Empty	Preventive	Calls	Broken	Empty	Preventive	Calls	Broken	Empty	Preventive	Calls	Broken	Empty	Preventive	Calls	Broken
Policy1	Spring	0.00%	20.28%	73.06%	6.67%	0.00%	74.72%	10.83%	14.44%	0.00%	16.67%	68.89%	14.44%	0.00%	3.33%	85.19%	11.48%
	Summer	0.00%	21.20%	70.92%	7.88%	0.00%	70.92%	14.95%	14.13%	0.00%	14.13%	72.83%	13.04%	0.00%	6.88%	78.99%	14.13%
	Autumn	6.87%	0.00%	83.79%	9.34%	43.41%	0.00%	42.31%	14.29%	3.30%	0.00%	85.71%	10.99%	0.00%	0.00%	90.11%	9.89%
	Winter	0.00%	68.75%	19.57%	11.68%	0.00%	60.05%	26.09%	13.86%	0.00%	69.57%	17.39%	13.04%	0.00%	60.14%	27.17%	12.68%
Policy2	Spring	0.00%	21.67%	71.67%	6.67%	0.00%	75.00%	10.56%	14.44%	0.00%	17.78%	68.89%	13.33%	0.00%	1.11%	88.52%	10.37%
	Summer	0.00%	18.75%	72.55%	8.70%	0.00%	71.47%	14.40%	14.13%	0.00%	19.57%	67.39%	13.04%	0.00%	3.99%	81.88%	14.13%
	Autumn	0.00%	18.68%	73.90%	7.42%	0.00%	43.13%	42.58%	14.29%	0.00%	19.78%	70.33%	9.89%	0.00%	12.09%	76.19%	11.72%
	Winter	0.00%	57.34%	30.43%	12.23%	0.00%	60.33%	24.73%	14.95%	0.00%	53.26%	36.96%	9.78%	0.00%	49.28%	36.59%	14.13%
Policy3	Spring	0.00%	71.11%	24.44%	4.44%	0.00%	72.50%	13.06%	14.44%	0.00%	71.11%	18.89%	10.00%	0.00%	71.11%	23.70%	5.19%
	Summer	0.00%	71.74%	24.73%	3.53%	0.00%	71.74%	14.13%	14.13%	0.00%	71.74%	20.65%	7.61%	0.00%	71.74%	22.10%	6.16%
	Autumn	0.00%	71.43%	25.27%	3.30%	0.00%	71.43%	19.23%	9.34%	0.00%	71.43%	24.18%	4.40%	0.00%	71.43%	23.44%	5.13%
	Winter	0.00%	70.65%	23.91%	5.43%	0.00%	70.65%	22.28%	7.07%	0.00%	70.65%	23.91%	5.43%	0.00%	70.65%	19.57%	9.78%
Policy4	Spring	0.00%	3.89%	90.28%	5.83%	0.00%	71.94%	13.61%	14.44%	0.00%	5.56%	82.22%	12.22%	0.00%	0.00%	94.81%	5.19%
	Summer	0.00%	0.00%	92.39%	7.61%	0.00%	69.84%	16.03%	14.13%	0.00%	0.00%	89.13%	10.87%	0.00%	0.00%	87.32%	12.68%
	Autumn	0.00%	1.10%	91.48%	7.42%	0.00%	34.34%	51.37%	14.29%	0.00%	0.00%	91.21%	8.79%	0.00%	0.00%	87.91%	12.09%
	Winter	0.00%	61.96%	23.91%	14.13%	0.00%	56.52%	29.08%	14.40%	0.00%	58.70%	27.17%	14.13%	0.00%	51.45%	35.51%	13.04%

Figure 16: How different policies use their time in recovery from very bad initial conditions

risk decreases: policy 2*** and policy 4*** reduce risk on average by about 95% and 91%, respectively. Furthermore, we see that these optimal information policies do not require more time to be spent on corrective actions: rather, breakages are addressed in a more timely manner. Figure 18 shows the very similar time usage of policy 2*** and policy 4*** compared to policy 2 and policy 4 (see Fig. 12).

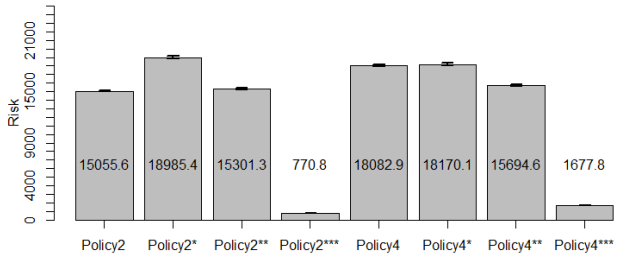


Figure 17: The average risk of applying policy 2 and 4 in stable state with 4 assumptions. Error bars show 95% confidence intervals on each mean. Policy 2 and policy 4 are running in the real-world scenario; policy2* and policy4* assume we do not have risk impact information; policy2** and policy4** assume there are no parking issues during preventative maintenance; policy2*** and policy4*** assume we can know any problematic gully pot immediately.

5. Conclusion

This paper has considered a real-world large-scale geographically-distributed maintenance prob-

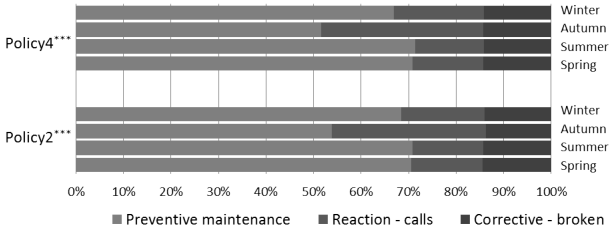


Figure 18: Time usage of policy2*** and policy4***

lem. It originates from the problem of gully-pot maintenance in the city of Blackpool, UK. The general aim is to reduce the overall surface water flooding risk caused by clogged gully pots in continuous time. The problem is modelled as a periodic travelling salesman problem (PTSP) and we propose a short period rolling planning approach that is able to automate adaptation to any environment changes. Unlike the standard PTSP, the main objective in our model is to minimize the risk measured by risk impact and failure rate of gully pots during a planning horizon. At the same time, all scheduled routes should be minimized in distance.

Due to the dynamic and large-scale features of our problem, we introduce a data structure (see Figure 4) for dealing with different types of actions. In addition, our objective function is highly sensitive to the gully pots' changing failure rates. We presented a hyperheuristic framework embed-

ded with a group of route and schedule-related low-level moves. This structure allows dynamic balancing between route and schedule optimization.

By adjusting different types of actions in different scenarios, our predictive scheduling strategy successfully out-performs the current real-world gully-pot maintenance approach, which is widely used in the UK, in terms of overall risk management, agility to react to emerging events, and robustness to poor initial states.

This predictive strategy relies significantly on the understanding of asset failure behaviour. Our estimation are based on working with experts in the field to provide the best (limited) data available. We are also working with Gaist Ltd. on a new surveying methodology which will further improve data in the future, but such data will not be available for some time to come.

We also estimate the potential investment to improve the maintenance performance. Results show the major reason for the current maintenance' poor performance is the latency of gully pot status information. Any technique that could accurately monitor the system status might afford an average 91% risk decreases every day, which would be worth an estimated £16,000 per day in Blackpool alone.

In further work we will investigate other investment possibilities. It is worth noting that work that Gaist Solution Ltd. has done to date on road maintenance decision support has resulted in investment worth hundreds of millions of pounds across several UK local councils.

6. Acknowledgements

The authors would like to thank Gaist Solutions Ltd. for providing data and domain knowledge. This research is part of the LSCITS project funded by Engineering and physical sciences research council (EPSRC).

References

D. Butler, Y. Xiao, S. Karunaratne, The gully pot as a physical, chemical and biological reactor, Water Science and Technology 31 (7) (1995) 219–228.

K. Karlsson, M. Viklander, Polycyclic aromatic hydrocarbons (PAH) in water and sediment from gully pots, Water, Air, and Soil Pollution 188 (1-4) (2008) 271–282, ISSN 00496979.

K. Scott, Investigating Sustainable Solutions for Roadside Gully Pot Management, Ph.D. thesis, 2012.

BBC, Flooding affects Pembrokeshire residents and businesses. Date accessed: 2015-09-04. , URL http://www.bbc.co.uk/news/uk-wales-15441912, 2011.

BBC, Floods: North Wales Police travel warning after rain. Date accessed: 2015-09-05. , URL http://www.bbc.co.uk/news/uk-wales-19702806, 2012.

Shieldsgazette, Flooding hell caused by blocked gully'. Date accessed: 2015-08-24., URL http://www.shieldsgazette.com/news/local-news/flooding-hell-caused-by-blocked-gully-1-4761698, 2012.

Leylandguardian, Blocked drain causes flooding danger on Lancashire road. Date accessed: 2015-08-24., URL http://www.leylandguardian.co.uk/news/blocked-drain-causes-flooding-danger-on-lancashire-road-1-7425326, 2015.

N. Christofides, J. E. Beasley, The period routing problem, Networks 14 (2) (1984) 237–256, ISSN 00283045.

V. C. Hemmelmayr, K. F. Doerner, R. F. Hartl, A variable neighborhood search heuristic for periodic routing problems, European Journal of Operational Research 195 (3) (2009) 791–802, ISSN 03772217.

D. Gulczynski, B. Golden, E. Wasil, The period vehicle routing problem: New heuristics and real-world variants, Transportation Research Part E: Logistics and Transportation Review 47 (5) (2011) 648–668, ISSN 13665545.

R. Baldacci, E. Bartolini, a. Mingozzi, a. Valletta, An Exact Algorithm for the Period Routing Problem, Operations Research 59 (1) (2011) 228–241, ISSN 0030-364X.

T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Operations Research 60 (3) (2012) 611–624.

S. Remde, K. Dahal, P. Cowling, N. Colledge, Binary exponential back off for tabu tenure in hyperheuristics, Evolutionary Computation in Combinatorial Optimization (2009) 109–120ISSN 03029743.

S. Duffuaa, M. Ben-Daya, K. Al-Sultan, a.a. Andijani, A generic conceptual simulation model for maintenance systems, Journal of Quality in Maintenance Engineering 7 (3) (2001) 207–219, ISSN 1355-2511.

R. Ahmad, S. Kamaruddin, An overview of time-based and condition-based maintenance in industrial application, Computers & Industrial Engineering 63 (1) (2012) 135–149.

A. H. Tsang, Condition-based maintenance: tools and decision making, Journal of Quality in Mainte-

nance Engineering 1 (3) (1995) 3–17, ISSN 1355-2511.

P. A. Scarf, C. A. V. Cavalcante, Hybrid block replacement and inspection policies for a multi-component system with heterogeneous component lives, European Journal of Operational Research 206 (2) (2010) 384–394, ISSN 03772217.

J. Wu, T. S. Adam Ng, M. Xie, H. Z. Huang, Analysis of maintenance policies for finite life-cycle multi-state systems, Computers and Industrial Engineering 59 (4) (2010) 638–646, ISSN 03608352.

M. C. Carnero Moya, The control of the setting up of a predictive maintenance programme using a system of indicators, Omega 32 (1) (2004) 57–75, ISSN 03050483.

J. Campos, Development in the application of ICT in condition monitoring and maintenance, Computers in Industry 60 (1) (2009) 1–20, ISSN 01663615.

P. Cowling, M. Johansson, Using real time information for effective dynamic scheduling, European Journal Of Operational Research 139 (2) (2002) 230–244, ISSN 03772217.

J. P. Kenne, L. J. Nkeungoue, Simultaneous control of production, preventive and corrective maintenance rates of a failure-prone manufacturing system, Applied Numerical Mathematics 58 (2) (2008) 180–194, ISSN 01689274.

F. Blakeley, B. Bozkaya, B. Cao, W. Hall, J. Knolmajer, Optimizing periodic maintenance operations for Schindler Elevator Corporation, Interfaces 33 (1) (2003) 67–79, ISSN 00922102.

W. Jang, H. H. Lim, T. J. Crowe, G. Raskin, T. E. Perkins, The missouri lottery optimizes its scheduling and routing to improve efficiency and balance, Interfaces 36 (4) (2006) 302–313, ISSN 00922102.

Y. J. An, Y. D. Kim, B. J. Jeong, S. D. Kim, Scheduling healthcare services in a home healthcare system, Journal of the Operational Research Society 63 (11) (2012) 1589–1599, ISSN 0160-5682.

P. Maya, K. Sörensen, P. Goos, A metaheuristic for a teaching assistant assignment-routing problem, Computers and Operations Research 39 (2) (2012) 249–258, ISSN 03050548.

J. Alegre, M. Laguna, J. Pacheco, Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts, European Journal of Operational Research 179 (3) (2007) 736–746, ISSN 03772217.

H. Tang, E. Miller-Hooks, R. Tomastik, Scheduling technicians for planned maintenance of geographically distributed equipment, Transportation Research Part E: Logistics and Transportation Review 43 (5) (2007) 591–609, ISSN 13665545.

I. García, J. Pacheco, A. Alvarez, Optimizing routes and stock, Journal of Heuristics 19 (2) (2013) 157–177, ISSN 1381-1231.

K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197, ISSN 1089778X.

I.-M. Chao, B. L. Golden, E. Wasil, An improved heuristic for the period vehicle routing problem, Networks 26 (6) (1995) 25–44.

J.-F. Cordeau, M. Gendreau, G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks 30 (2) (1997) 105–119, ISSN 0028-3045.

M. Laguna, R. Marti, Scatter search: methodology and implementations in C, Springer Science & Business Media, 2012.

C. Delgado, M. Laguna, J. Pacheco, Minimizing labor requirements in a periodic vehicle loading problem, Computational optimization and applications 32 (3) (2005) 299–320.

S. Pirkwieser, G. R. Raidl, Multilevel variable neighborhood search for periodic routing problems, in: P. Cowling, P. Merz (Eds.), Evolutionary Computation in Combinatorial Optimization, Springer Berlin Heidelberg, 226–238, 2010.

J.-F. Cordeau, M. Maischberger, A parallel iterated tabu search heuristic for vehicle routing problems, Computers & Operations Research 39 (9) (2012) 2033–2050, ISSN 03050548.

A. Rahimi-Vahed, T. G. Crainic, M. Gendreau, W. Rei, Fleet-Sizing for Multi-Depot and Periodic Vehicle Routing Problems Using a Modular Heuristic Algorithm, Computers & Operations Research 53 (2012) 0–27.

S. L. Cutter, S. Carolina, B. J. Boruff, Social Vulnerability to Environmental Hazards, Social Science Quarterly 84 (2) (2003) 242–261, ISSN 00384941.

UK GOV, Price Paid Data 2015. Date accessed: 2015-09-09., URL `https://www.gov.uk/government/statistical-data-sets/price-paid-data-downloads`, 2015.

S. A. Changnon, Record Flood-Producing Rainstorms of 1718 July 1996 in the Chicago Metropolitan Area. Part III: Impacts and Responses to the Flash Flooding, Journal of Applied Meteorology 38 (3) (1999) 273–280, ISSN 0894-8763.

A. Thieken, V. Ackermann, F. Elmer, H. Kreibich, B. Kuhlman, U. Kunert, H. Maiwald, B. Merz, M. Muller, K. Piroth, J. Schwarz, R. Schwarze, I. Seifert, J. Seifert, Methods for the evaluation of direct and indirect flood losses, 2008.

R. Brouwer, R. Van Ek, Integrated ecological, economic and social impact assessment of alternative flood control policies in the Netherlands, Ecological Economics 50 (1-2) (2004) 1–21, ISSN 09218009.

B. Merz, H. Kreibich, a. Thieken, R. Schmidtke, Estimation uncertainty of direct monetary flood damage to buildings, Natural Hazards and Earth System Science 4 (1) (2004) 153–163, ISSN 16849981.

Blackpool, Blackpool strategic flood risk assessment, Tech. Rep. June, URL `https://www.blackpool.gov.uk/Residents/Planning-environment-and-community/Documents/Blackpool-Strategic-Flood-Risk-Assessment.pdf.Dateaccessed:2015-07-05`, 2009.

W. Weibull, A statistical distribution function of wide applicability, Journal of applied mechanics 18 (1951) 293–297.

C. E. Ebeling, An introduction to reliability and maintainability engineering, Tata McGraw-Hill Education .

G. Clarke, J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12 (1964) 568–582.

N. Mladenović, P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (1997) 1097–1100.

P. Hansen, N. Mladenović, J. A. Moreno Pérez, Variable neighbourhood search: Methods and applications, Annals of Operations Research 175 (1) (2010) 367–407, ISSN 02545330.

S. Lin, Computer Solutions of the Traveling Salesman Problem, Bell System Technical Journal 44 (10) (1965) 2245–2269.

F. Glover, M. Laguna, Tabu Search, in: P. M. Pardalos, D.-Z. Du, R. L. Graham (Eds.), Handbook of Combinatorial Optimization, 3261–3362, 2013.

S. Remde, P. Cowling, K. Dahal, N. Colledge, E. Selensky, An empirical study of hyperheuristics for managing very large sets of low level heuristics, Journal of the Operational Research Society 63 (3) (2012) 392–405, ISSN 0160-5682.

P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, Practice and Theory of Automated Timetabling III (2001) 176–190.

E. Burke, B. MacCloumn, A. Meisels, S. Petrovic, R. Qu, A Graph-Based Hyper Heuristic for Timetabling Problems (2010a) 1–34.

R. Bai, J. Blazewicz, E. K. Burke, G. Kendall, B. McCollum, A simulated annealing hyper-heuristic methodology for flexible decision support, 4or 10 (1) (2012) 43–66, ISSN 16194500.

P. Garrido, M. C. Riff, DVRP: A hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic, Journal of Heuristics 16 (6) (2010) 795–834, ISSN 13811231.

M. Misir, A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete, Proceedings of the 9th Proceedings of the 9th Metaheuristic International Conference (2011) 1–10.

J. D. Walker, G. Ochoa, M. Gendreau, E. K. Burke, Vehicle routing and adaptive iterated local search within the HyFlex hyper-heuristic framework, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7219 LNCS (2012) 265–276, ISSN 03029743.

E. Özcan, B. Bilgin, E. Korkmaz, A comprehensive analysis of hyper-heuristics, Intelligent Data Analysis 12 (1) (2008) 3–23, ISSN 1088-467X.

K. Chakhlevitch, P. Cowling, Hyperheuristics: Recent developments, 2008.

E. Burke, M. Hyde, G. Kendall, A classification of hyper-heuristic approaches, Handbook of metaheuristics (2010b) 1–21.

E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art, Journal of the Operational Research Society 64 (12) (2013) 1695–1724, ISSN 0160-5682.