Railway Platform Reallocation After Dynamic Perturbations Using Ant Colony Optimisation

Jayne Eaton
Centre for Computational Intelligence (CCI)
School of Computer Science and Informatics
De Montfort University
The Gateway, Leicester LE1 9BH, UK
Email: jayne.eaton@dmu.ac.uk

Shengxiang Yang
Centre for Computational Intelligence (CCI)
School of Computer Science and Informatics
De Montfort University
The Gateway, Leicester LE1 9BH, UK
Email: syang@dmu.ac.uk

***Abstract*—Train delays at stations are a common occurrence in complex, busy railway networks. A delayed train will miss its scheduled time slot on the platform and may have to be reallocated to a new platform to allow it to continue its journey. The problem is a dynamic one because while reallocating a delayed train further unanticipated train delays may occur, changing the nature of the problem over time. Our aim in this study is to apply ant colony optimisation (ACO) to a dynamic platform reallocation problem (DPRP) using a model created from real-world train schedule data. To ensure that trains are not unnecessarily reallocated to new platforms we introduce a novel best-ant-replacement scheme that takes into account not only the objective value but also the physical distance between the original and the new platforms. Results showed that the ACO algorithm outperformed a heuristic that places the delayed train in the first available time-slot and that this improvement was more apparent with high-frequency dynamic changes.**

I. Introduction

The recovery of train schedules after a delay is an important factor in providing a reliable, cost-effective and better performing railway service. A train delayed when it arrives at a station will miss its scheduled time slot on its platform and may have to be reallocated to allow it to continue its journey. While reallocating the delayed train, further trains will be arriving at the station. If these are also delayed, they will change the platform allocation problem to a new problem with a different set of train arrival times. This makes the problem a dynamic one that changes over time. The aim of this research is to investigate the ability of ant colony optimisation (ACO), specifically the max-min ant system (MMAS) [17], to solve this dynamic platform reallocation problem (DPRP).

To investigate the DPRP, a model of a UK railway station, in this case Leicester station, has been created using Network Rail's schedule data from the Integrated Train Planning System (ITPS) [14]. The model details both the movement of trains through the station and the movement of all trains at each of the timing points on the trains' routes. This allows the long-term consequences of the reallocation decisions to be determined.

MMAS has no inbuilt intelligence to persuade it against unnecessarily reallocating trains to new platforms. Therefore, we also investigate a novel best-so-far ant replacement scheme that discourages this behaviour and minimises the unnecessary movement of trains to new platforms.

To evaluate the effectiveness of our approach, we compare the results with a heuristic that is used by railway controllers, that of assigning the delayed train to the first free platform (FFP), near to the original platform, that is available when it arrives at the station.

II. Related work

Previous work on the allocation of trains to station platforms using evolutionary computation (EC) techniques has been mainly concerned with scheduling rather than rescheduling trains. Ghoseiri and Morshedsolouk [8] used ant colony system (ACS) to schedule trains through several stations, where each station was connected by a single track. Conflicts were resolved by lengthening a train's dwell time and their aim was to minimise the overall increase in the dwell time. They found that ACS was comparable with an exact computation method and gave considerable time savings. Unfortunately, the work was limited by the fact that they used a very simplified railway model. Clarke *et al.* [4] used a genetic algorithm (GA) to reallocate trains to platforms at Glasgow Central station. They found that their system could successfully allocate around 1000 trains in approximately 30 seconds, with minimal clashes between scheduled trains.

Carey and Carville [1] applied a heuristic commonly used by train planners to a platform reallocation problem at Leeds station. Their heuristic sequentially resolved the conflicts for one train at a time. Their system not only produced a conflict-free timetable but could also measure the quality of the timetable in terms of factors such as platform usage and the number of platform adjustments. Carey and Crawford [2] extended the work to consider allocating trains to station platforms on a network of stations. They modelled 25 interconnected stations linked by multiple one-way lines in each direction. They based the problem on draft timetables and used the same heuristic as [1]. To extend the problem, they considered not only the station conflicts but also the conflicts on the lines leaving the station. They found that the algorithm could schedule the trains effectively in a reasonable time, however, the fact that they considered each station in

succession meant that the knock on delays increased with each successive station.

All of the above work is concerned with scheduling at train stations rather than rescheduling to recover the timetable after a disruption. Rescheduling trains is a popular research area, but seldom focuses on the issue of reallocating delayed trains at stations. For example, Khan *et al.* [12] used a GA to reschedule trains on a simulated single track railway section and found that they could produce a solution that reduced the train delay from 35 minutes to 12 minutes. Ho and Yeung [10] encoded a GA to tackle the problem of creating a feasible sequence of train to pass through a junction to minimize conflict after a train delay. They found that their algorithm could produce a solution within less than 5% of the optimal. Fan *et al.* [7] used both a GA and an ACO algorithm to resolve delays at a junction with good results while Chen *et al.* [3] used a modified differential evolution GA to tackle the problem of rescheduling trains after a delay at the St Pancras Midland Road Junction. The GA performed significantly better, in terms of minimizing passenger delays, than a first come first served (FCFS) heuristic.

The above research shows the potential of EC techniques in railway rescheduling trains after perturbations. However, in every case, the problem considered is a static one. In the real world, a train delay does not always occur in isolation. In many perturbed situations, there may be multiple delays and each new delay may change the nature of the problem, making it a dynamic one that changes over time. Work has started in the area of dynamic train rescheduling problems. Eaton and Yang [6] modelled a dynamic rescheduling problem on a junction of the UK railway network. They found that a population-based ACO (PACO) [9] algorithm outperformed a FCFS heuristic when the changes were frequent and of high magnitude. However, this work again concentrated on only a small section of the railway system. The station model used in this paper allows us to consider a much larger area of the railway network and to investigate the ongoing impact of reallocation decisions at a station.

III. The Dynamic Platform Reallocation Problem (DPRP)

A. Description of the Problem

The DPRP is the problem of reallocating multiple, successive, delayed trains to new time-slots at a railway station with the aim of minimising the ongoing delay in the system. In this work, we concentrate on reallocating trains at Leicester station. Leicester is a busy, bottleneck station with both passenger and freight trains coming from four different directions [11]. It contains four, bidirectional, passenger platforms with trains able to enter and leave any platform from any adjoining track section.

We consider the effect of the train reallocation not only on the trains at the station but also at all the timing points on the remainder of each train's journey within a specified radius of the station. In this way, we are able to take into account both the immediate and future outcome of the platform reallocations. The radius we consider is 50 miles (80.47 km) of the station which covers approximately 225 timing points.

Rescheduling trains at a station can be considered a two stage problem: the first stage is to find a place on the platform to place the delayed train, the second is to decide the order of trains to leave the station. In this paper, we consider only the first part of the problem, that of finding an efficient and feasible platform to place the delayed trains. In this work, trains leave the station in the departure order determined by their updated departure time. The second part of the problem will be tackled in our future work.

B. Mathematical Model for the DRRP

Each train (t) in T, the set of all trains, has a route that consists of a list of timing points and the times it arrives and departs each point. For each timing point r, in the set of all timing points in the problem (TP), there is a list of events (E_r) with each event corresponding to the arrival and departure of a train at that timing point. A train may be associated with more than one event at a timing point if it makes a return journey.

A track section is a length of track between two timing points. A timing point may be associated with more than one track section. At non-station timing points there will be two track sections, that are traversed in opposite directions. However, at a station, there will be several track sections, each corresponding to a platform. B_r is the set of track sections for timing point r.

The problem involves three decision variables; P_k, the platform assigned to event k for train t at the station, x_k^{arrive} and x_k^{depart} the arrival and departing times of event k associated with train t at the station and at each of the timing points on its ongoing journey. Table I describes the relevant notations used in the mathematical model.

1) Soft Constraints: To reduce disruption to passenger's journeys it is desirable to reallocate a delayed train to a platform close to the original platform. Constraint (1) expresses this, $posP_k^{original}$ and $posP_k^{new}$ are the physical positions of the original and the new platforms respectively. It is a soft constraint because although desirable it is not essential to satisfy it for the safe operation of the railway.

$$min(posP_k^{original} - posP_k^{new}) \quad \forall k \in E_s \tag{1}$$

2) Hard Constraints: It is essential to satisfy hard constraints for the safe and efficient running of the railway.

$$x_{k,b}^{depart} < x_{k+1,b}^{arrive} \quad b \in B_r, k \in E_r : k \neq l_{k,b} \tag{2}$$

Constraint (2) ensures the end time of an event at timing point r on track section b is less than the start time of the next event on that track section. This determines that there is no overlap between trains occupying the same track section and thus that only one train can occupy a track section at one time.

$$x_{k,b}^{depart} \geq d_{k,b}^{initial} \quad b \in B_r, k \in E_r \tag{3}$$

Constraint (3) ensures that the new departure time of event k on track section b is not earlier than the original departure time

TABLE I
NOTATIONS AND DESCRIPTIONS FOR THE MATHEMATICAL MODEL

Notation	Description
T	is the set of all trains in the evaluation window
t	is the index of a train in the evaluation window, $t \in T$
TP	is the set of all timing points in the problem
r	is the index of a timing point, $r \in TP$
B_r	is the set of all track sections, for one timing point r, $r \in TP$
b	is the index of a track section at a timing point, $b \in B_r$
E_r	is the set of all events at timing point r
E_s	is the set of all events at the station under investigation
k	is the index of an event in E_r, $k \in E_r$
$k+1$	is the first proceeding event of event k in E_r
$P_k^{original}$	is the original platform for event k, $k \in E_s$
P_k^{new}	is the new platform for event k, $k \in E_s$
P_k^{static}	is the forced platform for event k, $k \in E_s$
$a_{k,b}^{initial}$	is the initial scheduled arrival time of event k on track section b
$d_{k,b}^{initial}$	is the initial scheduled departure time of event k on track section b
$a_{k,b}^{static}$	is the forced arrival time of event k on track section b
$x_{k,b}^{arrive}$	is the reassigned arrival time of event k on track section b
$x_{k,b}^{depart}$	is the reassigned departure time of event k on track section b
$l_{k,t}$	is the last event k for train t, within radius and evaluation period, $k \in E_r$, $t \in T$
$l_{k,b}$	is the last event k for track section b, $k \in E_r$, $b \in B_r$
Z_t	represents the delay train t experiences when departing $l_{k,t}$
N	the number of station trains in the current evaluation window

specified in the train schedule. This constraint is especially important on the station platforms as trains that depart earlier than their scheduled departure time will cause chaos for passengers.

The start time (sp) of the problem is the time that the station controller is notified of the delayed train. Any trains that arrive before this time but depart after this time are transition trains that straddle the problem boundary. As they arrive before the start of the problem, they are not considered for reallocation to a new platform and their arrival time at the station is unchanged. However, their departure time at the station is included in the evaluation and may be changed. In addition their arrival and departure at the timing points on their ongoing journey may also be changed if they conflict with other trains wanting to use the same track section at the same time. Constraints (4) and (5) ensure that the platform and arrival time for transition trains remain unchanged at the station.

$$P_k^{new} = P_k^{static} \; k \in E_s : x_{k,b}^{arrive} < sp, x_{k,b}^{depart} \geq sp \quad (4)$$

$$x_{k,b}^{arrive} = a_{k,b}^{static} \; k \in E_s : x_{k,b}^{arrive} < sp, x_{k,b}^{depart} \geq sp \quad (5)$$

Trains that arrive and depart at the station before the station controller is notified of the delay remain unchanged in terms of allocated platform and arrival and departure at the station. However, if any timing points on their ongoing journey fall within the evaluation period then the arrival and departure at those timing points may be changed to remove conflict with other trains.

C. The Objective

When a train is delayed, it will miss its scheduled time-slot and its new arrival time may create conflict with other trains competing for the same resources. Conflict between two trains, train A and train B, can occur in two ways.

- Train A may arrive at a timing point before train B has left
- Train B may arrive at a timing point before train A has left

In this model, resolving the conflict involves delaying the arrival and departure time of the train that arrives second. The alternative would be to resolve the conflict by speeding up the departure of the first train. However, this would result in trains leaving the station before their scheduled departure time, which would cause disruption and frustration to passengers planning to use that service. The delay added to the second train, to resolve the conflict, is propagated through all of the timing points on the remainder of the train's route. As delaying a train may result in even more conflicts at subsequent timing points, the check for conflict is repeated until all the conflict has been removed from the system.

The objective is to minimise the delay of all the trains that pass through the station within the current evaluation window. The evaluation window determines how far into the future the controller wishes to assess the impact of the platform reallocation decisions. In these experiments it is set to one hour. A train's delay is calculated as in Eq. (6) and is the delay at its last timing point within the radius and within the evaluation window, whichever occurs soonest.

$$Z_t = x_{k,b}^{depart} - d_{k,b}^{initial} \quad k = l_{k,t} \quad (6)$$

The objective function is to minimise the delay of all station trains within the current evaluation window (Eq. (7)). Trains that terminate at Leicester station are not included in the evaluation as they have no ongoing journey.

$$min \sum_{t=1}^{N} Z_t \quad (7)$$

D. Construction of the Leicester Station Model

The station model was created from Network Rail's downloadable file of train schedules [14]. This feed is in JSON format and is an extract of train schedules from Network Rail's ITPS. It is freely available but requires registration with Network Rail. The use of this data ensures that we are only using information that is also available to the railway controller making the work suitable for contribution towards a computer-based dispatching system. The data feed contains details about all train schedules over a six-month time period. For each

train schedule, it provides an ordered list of the train's route, detailing the arrival and departure times at each timing point on its journey. In the UK railway, tracks are divided into blocks separated by signals. For the purpose of this work, we consider two timing points to sandwich a block section of track. Only one train can be in a block section at any one time [15, p71]. The assumption we have made in the creation of this model is that a train is said to depart its current block section once it has departed the timing point at the end of the block section.

The information needed to model the station's daily operation was extracted from each train's route. The direction the train travels through the station was determined by reference to the timing point the train passes as it leaves the station. However, we are not only interested in the movement of trains through the station but also the ongoing journey for each of those trains. Each timing point on a train's route may be used by other trains that may or may not pass through Leicester station. Therefore, the set of arrival and departure events occurring at each timing point on each train's route was also extracted from the data.

E. Modelling Dynamism

In an ideal world, we would have delay data available to investigate our approach. However, at this present time, Network Rail is unable to provide such data. Hence, we introduce delay to our model by delaying trains at specified time intervals by varying amounts. We model dynamism by adding successive delays at set time intervals. The time intervals between delays represents the frequency of change (f) while the length of delays represents the magnitude of change (m). As trains do not arrive at regular intervals, it is impossible to instigate a delay at an exact time, instead the train nearest to the start of the next change period is the one chosen to be delayed. A delayed train is not only delayed at the station but also at all its scheduled timing points after the station.

F. The First Free Platform (FFP) Comparison Heuristic

Discussions with a Network Rail Station Master established that a technique often used to reallocate delayed trains to platforms is to find the FFP as close as possible to the delayed train's original platform. The purpose of this is to minimise passenger and train crew disruption. We compare our ACO algorithm to a heuristic based on this principle. The first stage of the heuristic is to identify all possible gaps on the platform for the delayed train. These are gaps where the start time of the gap corresponds to the new arrival time of the delayed train and where the duration of the gap is large enough to accommodate the train's occupation of the platform track section. The constraints in Eqs. (8) and (9) need to be satisfied for a gap to be deemed a feasible gap for delayed train.

$$G_g^{start} = x_{k,b}^{arrive} \quad k \in E_r, b \in B_r \tag{8}$$

$$G_g^{duration} >= x_{k,b}^{arrive} - x_{k,b}^{depart} \quad k \in E_r, b \in B_r \tag{9}$$

where G is the set of all suitable gaps for the delayed train and g is the index of a gap ($g \in G$), G_g^{start} and $G_g^{duration}$ are the start time and duration of gap G_g respectively. A gap is also deemed suitable for a train if G_g^{start} is later than the train's delayed arrival time but $G_g^{duration}$ is large enough to cover the train's track occupation plus the extra delay needed to be added to the train to force it to arrive at G_g^{start}. In this case the train is delayed further, to allow it to fit into the gap, and the extra delay is propagated along all of the timing points on the train's route.

Once all suitable gaps are identified, FFP selects the gap on the platform that is closest to the train's original platform.

IV. ACO for Dynamic Optimization Problems (DOPs)

In ACO, ants communicate indirectly via pheromone trails. The pheromone trails hold the shared knowledge for the colony guiding them towards finding the optimal solution. If the ants make bad decisions, or if previous decisions become irrelevant to the environment, the redundant information is removed over time by evaporation of the pheromone trails. There are several variations of ACO algorithms [5]. In this work, we use the MAX-MIN AS (MMAS) algorithm developed by Stützle and Hoos [17] as it has a previous history of good performance.

A. Related Work Using ACO for Dynamic Rescheduling

The DPRP can be considered to be a dynamic job shop scheduling problem (JSSP). For a JSSP, the aim is to assign jobs to machines at particular time slots. For the DPRP, the jobs correspond to trains and the machines correspond to station platforms. Previous work in applying ACO to dynamic JSSPs has shown promising results. Both Xiang and Lee [18] and Renna [16] combined ant colony intelligence with a multi-agent system (MAS) to solve a dynamic JSSP. Xiang and Lee found that the MAS with ant colony intelligence outperformed a MAS with a first in first out (FIFO) dispatching rule, while Renna found that the ant intelligence approach gave a solution that was comparable with a coordination approach when the dynamic changes were of low or medium frequencies. Zhou *et al.* [19] applied ACO to a dynamic JSSP and found that it outperformed a heuristic based on the shortest processing time (SPT) while Lu and Romanowski [13] found that their version of ant colony system (ACS) outperformed well-known dispatching rules such as FIFO and SPT.

B. Max-Min Ant System (MMAS)

In this problem each node consists of a train and a feasible platform to allocate to that train (see Fig. 1). An ant's tour consists of a list of all trains within the problem with the platform assigned to them by the ant.

In MMAS, an ant, say ant k, when at node i, chooses the next node (train and platform combination) j in its neighbourhood N_i^k, probabilistically as follows:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{ if } j \in N_i^k, \tag{10}$$

where τ_{ij} is the pheromone information and η_{ij} is the heuristic information, α and β are constants which determine the

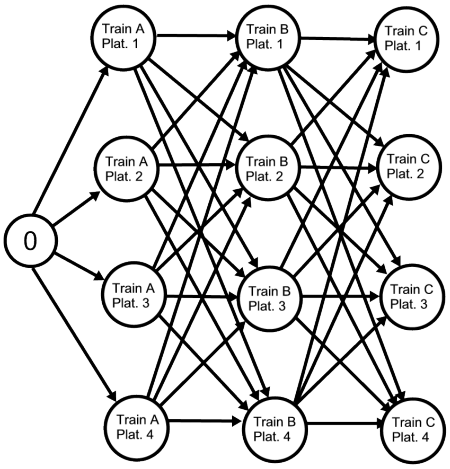

Fig. 1. The directed edge graph the ants use to construct their tour. Each circle represents a node.

relative influence of the pheromone and the heuristic values respectively.

After each iteration, all pheromone trails are evaporated as in Eq. (11).

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij},\ \forall(i,j) \in L, \tag{11}$$

where L is the set of all pheromones and $0 < \rho \leq 1$ is the pheromone evaporation rate, which is a constant parameter of the algorithm.

Then all pheromone trails are updated to correspond to the tour T^{best} of either the best-so-far ant or the best iteration ant as in Eq. (12).

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best},\ \forall(i,j) \in T^{best}, \tag{12}$$

The update value $\Delta\tau_{ij}^{best}$ is $\frac{1}{C}$, where C is the fitness of the best ant.

Two measures are employed to prevent stagnation. Firstly, the pheromone trails are bounded between a minimum τ_{min} and a maximum τ_{max} value, where $\tau_{max} = \frac{1}{C}$ and $\tau_{min} = \frac{\tau_{max}}{a}$ and a is a constant parameter of the algorithm. Secondly all trails are reset to τ_{max} when the algorithm shows stagnation behaviour or there has been no change in the best fitness for a set number of iterations.

C. Proposed MMAS Algorithm for the DPRP

To apply ACO to an optimization problem, it has to first be decomposed into a fully connected weighted graph $G = (V, E)$, where V is a set of vertices or nodes and E is a set of edges or connections between the nodes. In this case, each node represents a train on a platform. For example, node 1 represents train A on platform 1, node 2 represent train A on platform 2, etc (see Fig. 1). Ants move from node to node recording the nodes visited. Once they have selected a platform for one train they move on to the next train. At the end of their tour, they have constructed a solution that consists of a list of trains and the platforms they have been allocated to. At the end of the iteration, the best ant deposits pheromones on the edges of the graph, reinforcing its choices for the next iteration of ants. Ants are presented with trains in the order they arrive at the station. This is to prevent them from creating an infeasible solution in which a train is placed on a platform in front of a train that arrives before it.

After a delay some trains will have been removed from the problem as they will have arrived at the station and it will be too late to change their platform. These trains are no longer of interest to the ants. Therefore, they are removed from the directed edge graph that the ants move around to construct their tour, and are also removed from the pheromone matrix. However, new trains will be approaching the station and will be included in the new problem passed to the ants. These new trains are added both to the directed edge graph and to the pheromone matrix. Pheromone trails for all new trains are initialised to τ_{max}, however pheromone trails for trains that remain in the problem are retained to ensure that useful information from before the change is not lost.

Initial experimentation revealed that, as MMAS has no inbuilt intelligence to persuade it against unnecessary reallocation of trains to new platforms, the algorithm would often change a train's platform if the change had no impact on the fitness of the solution. In the real world, this would result in disgruntled passengers and unhappy railway employees. To solve this problem, we introduced a platform displacement heuristic (η_{ij} in Eq. (10)) to guide the ants in making intelligent platform reallocation choices. The heuristic takes into account the fact that Leicester station has two sets of two platforms separated by stairs. Platforms 1 and 2 are conjoined as are platforms 3 and 4. Moving between either of these is a simple case of crossing from one side of the platform to the other. However, moving from platform 1 or 2 to platform 3 or 4 involves negotiating a set of stairs. The heuristic ensures that the desirability of moving from stair-linked platforms is much lower than that of moving from conjoined platforms and that the desirability of both is much lower than leaving the train on its original platform. The heuristic is given by $1/PD$ where PD is a representation of the physical distance between the current node's platform and the decision node's platform. PD is 1 if the platforms are the same, 2 if they are conjoined, or 4 if they are separated by a set of stairs.

To reinforce the heuristic, we introduce a novel method to decide whether to replace the best-so-far ant (ant^{bs}) with the best iteration ant (ant^{bi}), after each iteration. This method takes into account both the solution's objective value and the amount of platform displacement. Experiments detailed in Section V-A show that this approach works well to reduce unnecessary platform reallocation.

V. Experimental Study

An experimental study was carried out to investigate the ability of our proposed MMAS algorithm to solve the DPRP. The parameters used for MMAS were established by preliminary experimentation. The best combination was found to be

TABLE II
DETAILS OF THE ALGORITHMS UNDER INVESTIGATION

Name	Heuristic Employed	Replacement Scheme
RS1-H	Yes	RS1
RS2-H	Yes	RS2
RS3-H	Yes	RS3
RS1	No	RS1

100 ants with $\rho = 0.5$, $a = 10$ and pheromone reinitialisation when there is no change in the best solution for 20 iterations. The best-so-far ant and the best iteration ant were used to update the pheromones in a ratio of 2:1 and both α and β were set to 1. The algorithm was run for 50 iterations before each dynamic change and 30 runs were executed for each dynamic scenario.

The first set of experiments is concerned with how effective the heuristic and the best-so-far ant replacement schemes are in reducing unnecessary train reallocations. The second looks at the effect of notification and planning interval on the performance of the algorithm. While the final set of experiments compares the performance of our MMAS algorithm to FFP's performance for each of the dynamic scenarios.

A. Comparison of Best-So-Far Ant Replacement Schemes

In this work, the best-so-far ant is the the best solution found since the beginning of the current dynamic change. Three best-so-far ant replacement schemes are investigated; RS1, RS2 and RS3. The details of each are given below.

- RS1: ant^{bs} is always replaced by ant^{bi} if ant^{bi}'s objective value (total delay) is less than ant^{bs}'s objective value
- RS2: ant^{bs} is only replaced with ant^{bi} if both the objective value (total delay) and the platform displacement for ant^{bi} is less than that for ant^{bs}
- RS3: ant^{bs} is always replaced with ant^{bi} if the objective value (total delay) for ant^{bi} is less than that for ant^{bs}. If the objective values for both ants are equal the amount of platform displacement is taken into account. If the platform displacement for ant^{bi} is less than that for ant^{bs}, ant^{bs} is replaced with ant^{bi}.

Table II summarises the combinations of the heuristic and replacement schemes investigated. Figures 2 and 3 show box plots of the average delay and platform displacement respectively for delay scenario $m = 20$, $f = 20$, for each replacement scheme. In each case, the delay and platform displacement are averaged over all dynamic changes. The horizontal line within the box represents the median, the top and bottom whiskers represent the maximum and the minimum values respectively, while the top and bottom of the box represent the third quartile and first quartile respectively.

With regards to platform displacement, RS2-H has the most positive effect on reducing platform displacement while RS1, which does not guide the algorithm in anyway towards reducing the number of platform changes, performs worse. RS1-H, which uses the platform displacement heuristic but

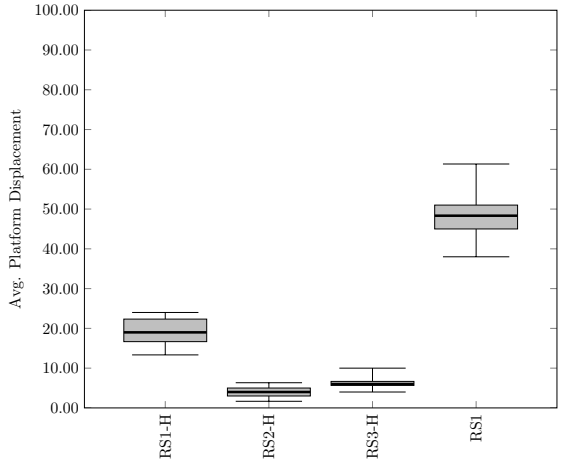

Fig. 2. Comparison of best-so-far ant replacement schemes for $m = 20$, $f = 20$

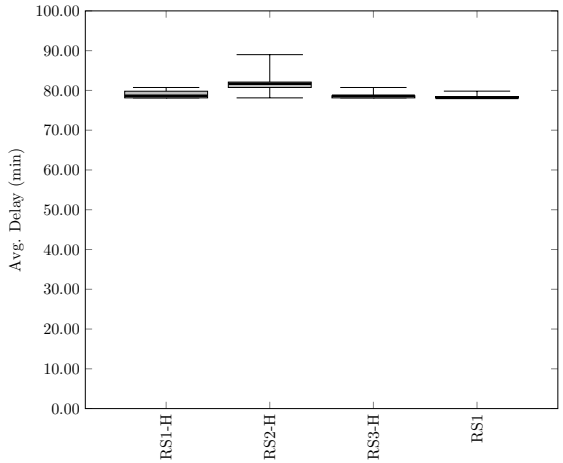

Fig. 3. Comparison of best-so-far ant replacement schemes for $m = 20$, $f = 20$

uses the original RS1 replacement scheme, also performs poorly on platform displacement compared to the algorithms that make use of either RS2 or RS3 replacment schemes. From Fig. 3, it is apparent that although RS2-H gives the best values in terms of platform displacement its effect may be too strong and its overemphasis on reducing the number of platform changes appears to restrict it in finding good results in terms of delay. For this reason, algorithm RS3-H is chosen to be implemented for the following experiments, as it provides a balance between reducing unnecessary platform changes and reducing delay.

B. Investigation into Notification and Planning Intervals

There are two interesting aspects of train delays from the viewpoint of the railway controller. The first is that of how much notification the controller has about the delay (the notification period), the second is that of how far into the future the controller wishes to look when considering where to place a delayed train (the planning horizon). To decide on

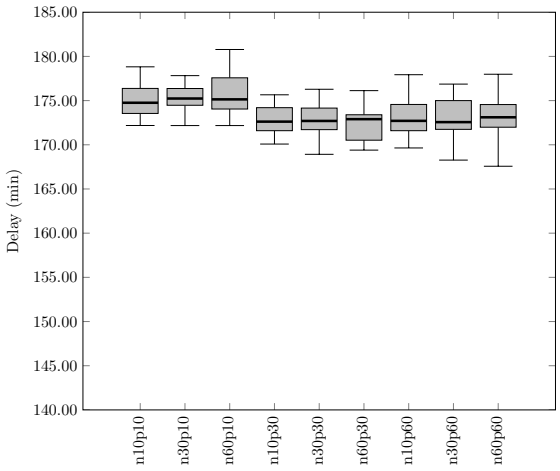

Fig. 4. Comparison of different notification and planning intervals for $m = 30$, $f = 10$

a notification period and planning horizon to use in our delay scenario experiments, we investigated different combination of notification periods and planning horizons. In this case, we looked at notification periods of 10, 30 and 60 minutes and planning horizons of 10, 30 and 60 minutes. Figure 4 shows the outcome of these experiments, where each experiment refers to a different notification and planning combination, for example, n10p60 refers to a notification period of 10 min and a planning horizon of 60 min.

Although a Kruskal-wallis test showed no significant difference between the medians of each notification/planning combination, we can see from the box plot that longer planning horizons of 30 min and 60 min give, on average, slightly less delay than a short planning horizon of 10 min. This is to be expected as the longer the planning horizon the more trains can be included in the problem given to the ants and the more options they have to rearrange those trains on the platforms to reduce the objective value. In contrast, the notification interval appears to have very little influence on the performance of the algorithm. For example, the first three boxes for p=10 and n=10, 30 and 60 all have similar median values. The longer the notification period, the more trains can be reshuffled before the delayed train arrives. This result suggests that rearranging trains that arrive before the delayed train has very little influence on the outcome of the algorithm and that it is trains that arrive later than the delayed train that are the important ones in terms of finding a good problem solution.

For our subsequent investigations, we chose a planning horizon of 60 mins to give the best possible outcome. As already mentioned the notification period appears to have very little influence on this problem and so a value of 30 min was chosen.

C. Delay Scenarios

Nine different dynamic environments were investigated involving all permutations of 3 different magnitudes of change (10 mins, 20 mins, 30 mins) and 3 different change frequencies (10 mins, 20 mins, 30 mins). All delays occurred over a period of one hour and started at 7am.

Figure 5 shows the results in terms of delay for MMAS compared with FFP for the scenarios with $m = 30$. The delay scenarios with $m = 20$ and $m = 10$ showed a similar pattern of results. The graphs indicate that MMAS is better at minimising delay than the FFP Heuristic. The difference is most pronounced for the scenario with the high frequency changes ($f = 10$) and the improvement in peformance becomes more apparent as more and more trains are delayed. This is because the MMAS algorithm has the freedom to rearrange all the trains on the platform to make the earliest possible space for the delayed trains whereas FFP must use the first space that becomes available, which may be later in time.

It is interesting that, for the 7.25am change when $f = 10$, FFP performs slightly better than MMAS. This may be because of the time-linked nature of this problem. In MMAS, the best-so-far solution from before the change is used to initialise the environment after a change. Thus, the solution found before the change restricts the reallocation options available after the change, which resulted in a slightly poorer performance for MMAS.

The results were tested for statistical significance using the non-parametric, two-tailed, one-sample Wilcoxon signed rank test at a 0.05 significance level. The one-sample test allows us to compare the results of 30 runs for the ACO algorithm with the single result from the FFP algorithm. The average delay over all the changes for FFP was compared with the average best-so-far delay over all the changes for MMAS. Table III gives the results of comparing MMAS $\Leftrightarrow$ FFP, where the symbol '$+$', '$-$' or '$\sim$' indicates that MMAS is significantly better than, significantly worse than, or not significantly different from FFP, respectively. The results reveal that in all scenarios MMAS performed significantly better than FFP.

The experiments were run on a 2.9GHz Intel Xeon E5-2666 v3 (Haswell) processor. An analysis of the computation time showed that, for small magnitude, low frequency changes, the algorithm took an average of 39.3 seconds to run while for high magnitude, high frequency changes, it took an average of 67.8 seconds to run. The high magnitude, high frequency scenarios have an increased execution time due to the fact that more trains are affected by the disruption, which increases the time needed to resolve all the conflicts. Nevertheless, for the system to obtain a solution in less than 70 seconds indicates its feasibility to be used in a real-world time-dependent situation.

VI. Conclusions and Future Work

Reallocating trains to platforms after a delay is a complex task, made more complicated by the fact that it can be a dynamic problem that changes over time as more delayed trains arrive at the station. In this work, we applied MMAS to the DPRP using a model created from real-world train schedule data for a busy UK railway station. The model allows us to project the outcome of the reallocation decisions into the future by accessing the impact on a train's ongoing journey

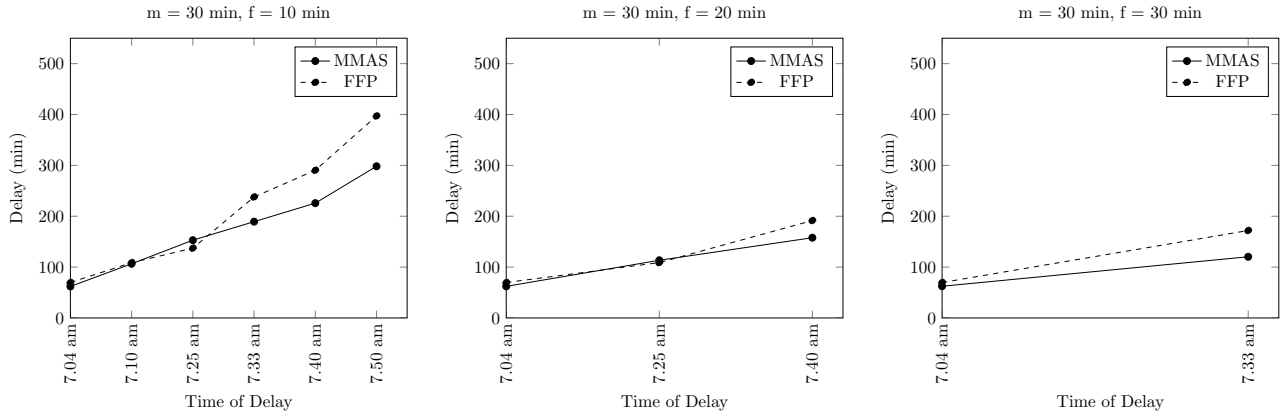

Fig. 5. Experimental results comparing the average Best-So-Far solution for MMAS with FFP for $m = 30$ and $f = 10, 20$, and 30.

TABLE III
ONE-SAMPLE WILCOXON SIGNED RANK TEST RESULTS AT A 0.05 SIGNIFICANCE LEVEL

	m=30			m=20			m=10		
Algorithms	f=10	f=20	f=30	f=10	f=20	f=30	f=10	f=20	f=30
MMAS $\Leftrightarrow$ FFP	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$

in terms of conflicts with other trains sharing the same block sections. Results showed that the ACO algorithm outperformed a heuristic that placed the delayed train in the first available time-slot and that this improvement was more apparent with high-frequency dynamic changes. In addition, the use of a platform displacement heuristic combined with a novel best-so-far ant replacement scheme worked to give the ants the intelligence to minimise unnecessary platform changes.

This research is seen as a step towards a system that could be implemented within a computer-based dispatching system to support the railway controller in solving schedule conflicts after a perturbation. It has stimulated a number of future research ideas. For example, we aim to apply the solution to a much more complex station, such as Birmingham New Street. This station has 12 platforms, each split into three usable subsections, which makes the spatial problem of reallocating trains to platforms much more difficult. We also aim to solve the second part of the platform reallocation problem by introducing an additional ACO algorithm that explicitly determines the sequence of trains to leave the station.

ACKNOWLEDGMENT

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/K001310/1.

REFERENCES

[1] Carey, M. and Carville, S.: Scheduling and platforming trains at busy complex stations. Transport. Res. Part A: Policy and Practice. 37(3), 195–224 (2003)

[2] Carey, M. and Crawford, I.: Scheduling trains on a network of busy complex stations. Transport. Res. Part B: Method. 41(2), 159–178 (2007)

[3] Chen, L., Schmid, F., Dasigi, M., Ning, B., Roberts, C. and Tang, T.: Real-time train rescheduling in junction areas. In: Proc. of the Institution of Mechanical Engineers, Part F: J. Rail and Rapid Transit, 224(6), 547–557 (2010)

[4] Clarke, M., Hinde, C. J., Withall, M. S., Jackson, T. W., Phillips, I. W., Brown, S. and Watson, R.: Allocating railway platforms using a genetic algorithm. In: Research and Development in Intelligent Systems XXVI, Springer, 421–434 (2010)

[5] Dorigo, M. and Stützle, T.: Ant Colony Optimization. Bradford Company, MA, USA (2004)

[6] Eaton, J., Yang, S., and Mavrovouniotis, M.: Ant colony optimization with immigrants schemes for the dynamic railway junction rescheduling problem with multiple delays. Soft Computing, 2015. [Online]. Available: http://dx.doi.org/10.1007/s00500-015-1924-x

[7] Fan, B. Roberts, C. and Weston, P.: A comparison of algorithms for minimising delay costs in disturbed railway traffic scenarios. J. Rail Transport Planning & Management, 2, 23–33 (2012).

[8] Ghoseiri, K. and Morshedsolouk, F.: ACS-TS: Train scheduling using ant colony system. Advances in Decision Sciences, vol. 2006, pp. 1–28 (2006)

[9] Guntsch, M. and Middendorf, M.: Applying population based ACO to dynamic optimization problems in Ant Algorithms. Springer, Berlin Heidelberg pp. 111–122 (2002).

[10] Ho, T. and Yeung, T.: Railway junction conflict resolution by genetic algorithm. Electr. Lett., 36(8), 771–772 (2000)

[11] Hopper, C.: Response to draft consultation document East Midlands route study, https://www.networkrail.co.uk/East-Midlands-Route-Study-Consultation-Responses.pdf [Accessed on 20th Jan 2016]

[12] Khan, M., Zhang, D., Jun, M. shi, and Li, Z. J.: An intelligent search technique to train scheduling problem based on genetic algorithm. In: Proc. 2006 Int. Conf. on Emerging Tech., pp. 593–598 (2006)

[13] Lu, M.-S. and Romanowski, R.: Multi-contextual ant colony optimization of intermediate dynamic job shop problems. Int. J. of Adv. Manuf. Tech., 60(5-8), 667–681 (2012)

[14] Network Rail, http://www.networkrail.co.uk/data-feeds/ [Accessed on 1st October 2015]

[15] Pachl, J.: Railway Operation and Control, 2nd ed. VTD Rail Publishing, Mountlake Terrace, USA (2009)

[16] Renna, P.: Job shop scheduling by pheromone approach in a dynamic environment. Int. J. Comput. Integ. Manuf., 23(5), 412–424 (2010)

[17] Stützle, T. and Hoos, H.: MAX-MIN ant system and local search for the traveling salesman problem. In: Proc. 1997 IEEE Int. Conf. Evol. Comput., pp. 309–314 (1997)

[18] Xiang, W. and Lee, H.: Ant colony intelligence in multi-agent dynamic manufacturing scheduling. Eng. Appl. Artif. Intell., 21(1), 73–85 (2008)

[19] Zhou, R., Nee, A., and Lee, H.: Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems. Int. J. Prod. Res., 47(11), 2903–2920 (2009)