A Mixed-Integer Programming Approach for Scheduling Roadworks in Urban Regions

Mauro Vallati[1][0000−0002−8429−3570] and Lukáš Chrpa[2][0000−0001−9713−7748]

[1] University of Huddersfield, Huddersfield, United Kingdom
[2] Artificial Intelligence Center, Czech Technical University in Prague, Czech Republic

Abstract. In order to keep roads in acceptable condition, and to perform maintenance of essential infrastructure, roadworks are required. Due to the increasing traffic volumes and the increasing urbanisation, road agencies are currently facing the problem of effective planning frequent –and usually concurrent– roadworks in the controlled region. However, there is a lack of techniques that can support traffic authorities in this task. In fact, traffic authorities have usually to rely on human experts (and their intuition) to decide how to schedule and perform roadworks. In this paper, we introduce a Mixed-Integer Programming approach that can be used by traffic authorities to plan a set of required roadworks, over a period of time, in a large urban region, by specifying constraints to be satisfied and suitable quality metrics.

Keywords: Mixed-Integer Programming · Roadworks · Urban Traffic Management.

1 Introduction

It is expected that, during the 21st century, there will be a huge growth in urbanisation as the global population living in urban areas is projected to rise to 66% by 2050 from 54% in 2014.[3] Together with the expected socio-economic motivation for increasing mobility, a significant increase of pressure on the urban traffic network seems to be inevitable. Given the increasingly demanding use of the urban traffic network, road maintenance works are becoming more frequent, extremely time-constrained, and of primary importance [3]. Furthermore, the continuous growth of the size of urban areas is requiring additional road works to be performed, in order to maintain in operation (or to improve) all the infrastructure needed by the growing urban population, e.g., electricity, gas, network cables, etc. As a result, on top of the other urban traffic management tasks, transport controllers are currently facing the problem of coping with the large amount of roadworks to be planned and performed at a regional level. Planning ahead is crucial, because the impact of roadworks can be dramatic on the traffic of the area. A notable example is represented by the roadworks required in the

[3] United Nations - http://esa.un.org/unpd/wup/Publications/Files/WUP2014-Highlights.pdf

Manchester (UK) area in 2018, which resulted in an almost daily traffic jams and, consequently, in dramatic disruptions to the local economy.[4]

Despite the pressing need for efficiency, and the large body of works for optimising problems and situations daily faced by traffic controllers, there is a lack of approaches designed for supporting agencies in the decision process of creating a plan of roadworks for a controlled urban area. Beside a very recent preliminary exploration of the use of AI planning for planning roadworks [14], the body of existing works in the area is focused on providing formal frameworks for the analysis of the impact of works [6], or on the evaluation of the process exploited by local authorities [7]. For this reason, roadwork plans are usually generated manually, without any explicit notion of "quality", and with a very limited control on constraints and assessment of the impact of works on the managed network traffic, as highlighted by the above-mentioned Manchester example.

In this paper we propose a principled approach that leverages Mixed-Integer Programming (MIP) [2] to provide an effective Decision Support tool for planning roadworks in urban areas. Mixed-Integer Programming provides a powerful framework for maximising an objective function subject to one or more constraints, where variables can have Boolean or Integer values. Examples of use of MIP in real-world applications include the generation of optimal scheduling for constellation observation [13], optimising the loading of boxes into air cargo containers [10], the generation of fiducial marker dictionaries [5], and the design of reliable chemical plants [15].

Here we provide a formalisation of the roadworks scheduling problem, and using the well-known MiniZinc tool [1], we specify and develop a MIP model that allows to represent the constraints related to roadworks planning within large urban regions, and to include notions of quality in order to shape the generated solutions. Notably, MiniZinc is able to translate MIP models, encoded in a human-readable language, in the FlatZinc language [9], that is supported by a wide range of solvers: solvers can therefore be used as "black-boxes" that can be easily embedded (and are easy to change and update) into a larger system. We empirically validate our approach by considering the Kirklees urban region of Yorkshire (UK), and by modelling scenarios based on roadworks performed in the region.

2 Problem Definition and Modelling

Roadworks are needed for three main purposes:

- (i) maintenance of the existing road network;
- (ii) extension or improvement of the network;
- (iii) works not strictly related with the traffic network

The third case includes works such as water or gas supply pipes maintenance. For the sake of this investigation, we focus on cases (i) and (iii), as they are the

[4] `https://bbc.in/2C6l1mu`

most common in large urban areas, and only temporarily affect the performance and the characteristics of the network. Case (ii) works are rarely put in place in well-established urban networks, and may require a specific planning and scheduling.

2.1 Assumption

We assume that a number of roadworks need to be performed in a controlled urban region. Each roadwork is characterised by:

- *Duration*: The length of time required to complete the work. This is fixed and known before optimising the roadworks schedule.
- *Deadline*: The latest available time in which the work can be completed.
- *Start*: The earliest available time in which the work can be started.
- *Location*: The location of the roadwork. Here the location can be specified in terms of a urban *area*, or in terms of the specific affected road(s). The number of works in a given area, or in adjacent areas has to be limited to mitigate traffic congestion issues.
- *Company*: The company that is in charge of performing the roadwork. We assume that each company has a maximum number of works that can carry out concurrently.

We focus on works that have already been identified and accepted, and that are ready to start in the near future. We assume that at schedule time the set of works and their information as specified above are known and fixed. Managing works then involves determining which roadwork can be performed, considering constraints on the number of works a company can perform at the same time and the number of works that can be done concurrently in a given urban area. The optimisation requires that works are scheduled as early as possible. We assume that time is suitably discretised.

2.2 Model

Let us introduce the different variables and sets used in our optimisation model. Let $W = \{w_1, ..., w_n\}$ denotes the set of works that are being scheduled, and let $B = \{b_1, ..., b_n\}$, $D = \{d_1, ..., d_n\}$, and $E = \{e_1, ..., e_n\}$ denote, respectively, the lists of Begin, Duration, and Deadline times for each work. The begin value provides the earliest time step in which the corresponding work can be started, duration refers to how many time steps the work requires, and deadline refers to the latest time step in which the work has to be finished. Intuitively, given a work w_i, the corresponding values are identified by b_i, d_i, and e_i. The following constraints illustrates the relationships between the values that has to met (otherwise a work violating those constraints cannot be scheduled).

$$\forall w_i : (1 \leq b_i \leq e_i) \tag{1}$$

$$\forall w_i : (d_i \leq (e_i - b_i)) \tag{2}$$

In particular, constraints 1 and 2 are used to encode aspects related to the timings of works to be scheduled, and to make sure that the problem at hand is feasible –at least from this perspective. In particular, Constraint 1 defines the relationship between begin and deadline time of each work, and at the same time it make sure that the earliest begin time is a positive number. Constraint 1 is used to enforce the fact that the time window which is allowed for a work is sufficient, according to the duration.

To represent an actual starting time of each of the works, we introduce the list $S = \{s_1, ..., s_n\}$, which is also the objective of the optimisation.

In this paper we consider that each time unit t_i corresponds to one week, but that can be easily adjusted without changes to the model. The week time discretisation has been selected because it allows some flexibility, with regards to works that are requiring a slightly longer amount of time than the initially budgeted.

We do not explicitly model roads, but focus instead on the notion of areas, which represent a group of connected roads. The notion of area allows to effectively and efficiently encode the proximity of roadworks, as discussed later in this section. Let $A = \{a_1, ..., a_m\}$ be the set of considered areas for the given urban region. Areas that are adjacent are identified by a predicate $nextTo(a_i, a_j)$. Each area has a maximum number of roadworks that can be concurrently performed, denoted by $maxWorks(a_i)$. For the sake of assessing the number of works performed at the same time, we introduce a counter $active(a_i, t_j)$, that is used to count how many works are planned to performed at time t_j in the area a_i.

Similarly, we would like to constrain the number of concurrent works performed by a single company. The maximum number of works that a company c_i can perform at the same time is defined via $maxWCompany(c_i)$. Works are assigned to a company using a predicate $assignedTo(w_i, c_j)$, while the number of active works of a given company is specified by the counter $activeWCom(c_i, t_j)$.

Combining the assumptions and constraints described above, we can then define the model used for scheduling a set of roadworks in a urban region:

$$\forall w_i : (s_i + d_i \leq e_i) \tag{3}$$

$$\forall w_i : (s_i \geq b_i) \tag{4}$$

$$\forall a_i, \forall t_j : (active(a_i, t_j) \leq maxWorks(a_i)) \tag{5}$$

$$\forall c_i, \forall t_j : (activeWcom(c_i, t_j) \leq maxWCompany(c_i)) \tag{6}$$

$$\begin{aligned} \forall t_j, \forall a_i, a_x : nextTo(a_i, a_x) \wedge \\ (active(a_i, t_j) < maxWorks(a_i) \wedge \\ (active(a_x, t_j) < maxWorks(a_x) \end{aligned} \tag{7}$$

```
set of int: WEEKS = 1..period;

constraint forall (w in WORKS)
    (schedule[w] => begin[w]);

constraint forall (a in AREA,
    b in AREA where nextTo[a,b] == true ) (
        forall (t in WEEKS)
            (active[b,t] < maxWorkArea[b] /\
             active[a,t] < maxWorkArea[a]));
```

Fig. 1. MiniZinc encoding of Constraints 4 (top) and 7 (bottom). The first line shows how the time period can be manually defined.

Constraints 3 and 4 are used to enforce the correct scheduling of a work, with regards to the deadline (3) and the earliest begin time (4).

Constraints 5 and 6 allow to encode limits related to the maximum works that can be active at the same time step t_i. Constraint 5 focuses on limits due to works being performed in the same area, while Constraint 6 is used to limit the number of works that a company has to perform concurrently. Finally, Constraint 7 is used to specify that it is not allowed to perform concurrently too many roadworks on adjacent areas. This has been encoded by constraining the fact at the same time, the maximum number of allowed works can not be performed in two neighbouring areas.

The mixed-integer problem is solved when all the required roadworks have been scheduled, via the dedicated list S, while respecting all the constraints of the model. The quality of a solution is calculated by considering the scheduled starting time of the works, and the function to optimise can be modelled as follows.

$$min \sum_{i=1}^{n} s_i \tag{8}$$

In other words, the goal is to schedule all the works as early as possible.

2.3 Implementation

We decided to implement the described Mixed-Integer Problem using the well-known MiniZinc [1] constraint modelling language. This is because MiniZinc provides a valuable middle ground for encoding problems. On one hand, it is high-level enough to express most constraint problems easily. On the other hand, it can be mapped onto existing solvers easily and consistently via the translator that allows to generate FlatZinc models.

The overall encoding of the above described constraints required a few tens of lines of code in MiniZinc. Figure 1 provides an example of the implementation

of two constraints, namely 4 (top) and 7 (bottom). While the vast majority of the code should be self-explanatory, there are a couple of caveats that should be described in order to support the readability. The first line is used to define the time period that is considered for the scheduling. The `period` value is provided as input, and represents the number of weeks to consider. This approach greatly simplifies the encoding of constraints that need to be enforced for each time step. Finally, the symbol indicates a *logic or* operator in the MiniZinc language.

3 Evaluation of the Approach

This section is devoted to evaluate the capability of the proposed MIP-based approach to effectively and efficiently generate multiple roadwork plans for a given urban region. The MIP model has been encoded using MiniZinc version 2.2.3. Generated problems have been solved using three general solvers: Chuffed 0.10.3[5], Gecode 6.1.0 [11], and CBC 2.9 [4]. The first solver is based on lazy clause generation, which is a hybrid approach that combines features of finite domain propagation and Boolean satisfiability. Gecode is the winner of all categories at the MiniZinc Challenges between 2008 and 2012 [12], and is extremely optimised for the FlatZinc language. Finally, CBC is a branch-and-cut solver that requires MIP models to the provided via the COIN-OR open solver interface.

In this context, quality is measured in terms of starting time of works, as indicated in Equation 8. In other words, works have to be scheduled as soon as possible. Better quality corresponds to works which, on average, are started as early as possible. This is the key indicator of quality, in the light of the fact that, in a typical urban context, all the roadworks which need to be planned are usually essential, and should be completed as soon as possible.

Experiments have been performed on a system equipped with 2.0 Ghz Intel i7-3667U Processors, 8 GB of RAM and Linux operating system. We set a cutoff time of 1 CPU-time minute (60 seconds). Due to the way in which considered solvers work, generated plans are guaranteed to be valid with regards to the provided MIP model.

Our analysis has been focused on the northern part of the Kirklees urban region. Kirklees is a council of the West Yorkshire county, located in the north-east part of United Kingdom. Here, 11 different areas has been identified, following the local classification and organisation: Ashrbow, Almondbury, Marsh, Lindley, Edgerton, Crosland Moor, Netherton, Dalton, Huddersfield centre, Lindley, and Newsome. For the sake of this analysis, here we considered only areas as a whole. It should be noted that roads can be easily added according to the network that needs to be modelled and to the level of granularity required. Figure 2 (coloured) shows an overview of the considered region.

3.1 Empirical Results

In order to collect informative and realistic data, we analysed the number of roadworks being executed in the considered region during 2017. According to the

[5] `https://github.com/chuffed/chuffed`

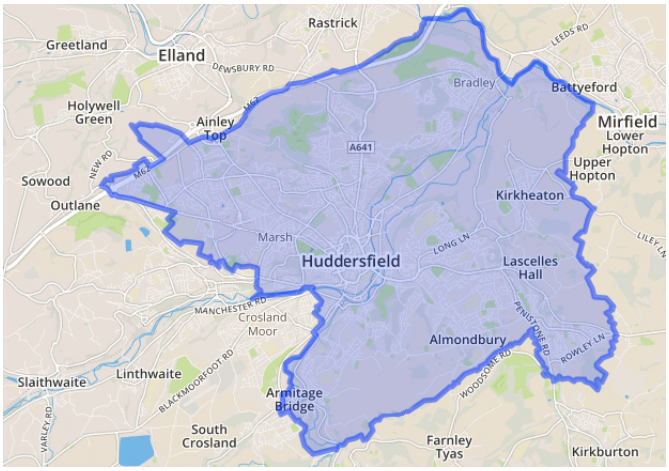

Fig. 2. The urban region considered in the experimental analysis: it includes 11 areas, such as Marsh and Huddersfield Centre, within the Yorkshire county of the United Kingdom.

observed data, we synthetically generated a set of instances, with the following main characteristics: number of roadworks to be planned ranging from 5 to 20; plan horizon of 6 months, roadworks duration ranging between 1 and 10 weeks; at most 2 roadworks can happen concurrently in the same area, but two nearby areas can not have, at the same time, the maximum amount of works scheduled. This is to avoid too much pressure on the network due to traffic navigating in those adjacent areas. Start time and deadline values have been randomly generated. For each considered number of roadworks to be planned, we randomly generated three instances with different durations and start times. As a general common deadline, we required all the works to be completed by the end of the modelled 6-months period. We considered the roadworks to be assigned evenly to 5 companies, and that each company can perform a single work at a time.

Presented results are averaged on the three instances, in order to take into account noise and variability due to the randomised aspects.

The results of this first set of experiments, designed for testing the fact that the proposed approach is effectively able to plan the roadworks required in the controlled region, are presented in Table 1. As a first remark, it is worth noting that all the considered solvers are able to generate solutions in a very reasonable amount of time. In fact, the process of planning roadworks is not required to be performed in real time, and longer amount of CPU time can be devoted to the task. Nevertheless, less than 1 CPU-time minute has been enough to generate solutions to all the considered cases. It should be noted that the considered solvers run in an "anytime" fashion: they keep running until they are not able to find a better quality solution to the considered problem. Furthermore, they

Table 1. Runtime (CPU-time seconds) needed by the considered solvers for providing a plan for performing an increasing number of roadworks in the considered area over a 6-month period of time. Quality of generated plans is measured as the average delay (time units–weeks) with regards to the first possible start time of the work. Bold indicates best results, * indicates that the solution is different from those of the other solvers.

	Runtime			**Quality**		
# works	5	10	20	5	10	20
Chuffed	0.3	0.4	**18.3**	2.2	2.8	**5.1**
Gecode	**0.2**	**0.3**	60.0	2.2	2.8	7.6
CBC	0.5	3.8	60.0	2.2	2.8*	5.7

also provide intermediate solutions found in the meanwhile; an interested user can therefore let the system run for a higher amount of time, and at the same time can have insights about the current best solution. Among the considered solvers, Chuffed is the one that is generally the fastest. CBC is possibly negatively affected by the fact that an additional translation step has to be performed in order to provide the solver with the required input format.

With regards to the quality of generated plans, the easiest tasks are solved optimally by all the considered solvers. As multiple plans can share the same overall quality value, due to the fact that the measured quality corresponds to an average delay among works to be started, it is interesting to notice that Chuffed and Gecode tend to return the same plans, while CBC seems to "prefer" plans where shortest works are scheduled as soon as possible, while longest works are delayed. On the contrary, the other considered solvers tend to schedule longest works earlier.

Figure 3 shows (top) how the output can be shaped using the MiniZinc language, and (bottom) part of an example output provided by the proposed approach. In the shown case, four roadworks are planned to start in different weeks –which correspond to the time unit of our model. The company that has to perform the work is also listed. The shape of the output can be modified according to needs, as the MiniZinc interface provide significant flexibility on this regards.

To check the validity of generated plans, they have been visually inspected by transport experts, which confirmed their overall quality and that they look similar to plans one would expect from an expert-generated solution.

3.2 Time horizons

To study the impact on performance of the number of roadworks to be planned, in the previous section we focused our analysis on scenarios with increasing numbers of roadworks, where time windows are randomly generated: the end time of all the works was fixed at the end of the 6-months period, but the different starting time lead to different time windows for each of the works. Intuitively, the size of the time window is an extremely important aspect: loose

```
output [ "\(assignedTo[w]) starts \(w) in \(schedule[w]);\n" | w in
WORKS ];
```

```
Company1 starts workA in week 5
Company2 starts workB in week 5
Company4 starts workC in week 11
Company1 starts workX in week 9
```

Fig. 3. Top: the command used in MiniZinc to format the provided output. Bottom: Excerpt of plan provided by the MiniZinc framework for planning four roadworks (namely, workA, workB,workC, and workX) in the controlled region. The provided week indicates the planned start time, and company to which the work has been assigned.

time windows give a significant degree of freedom to the solver to schedule works, while very strict time windows instead steadily drive the search process. Here we therefore aim to assess the impact of time windows on the performance of the proposed MIP-based approach. Focusing on the 10 roadworks scenario, we generated four problems by varying (i) the duration of roadworks, either 1 or 10 for all the considered works, and (ii) the size of the time window of each work, that can be exactly the same size of the duration, or can cover the whole modelled period.

As a first remark, we note that the duration of the considered works has no significant impact on the performance of the considered solvers: neither CPU-time nor the quality of the generated schedule plans is affected. On the other hand, the size of the time window has a remarkable impact on performance. The more constrained the time windows are, the easier it seems to be for the solvers to generate a solution. In the more constrained cases, all the 3 approaches can find a plan in approximately 0.3 CPU-time seconds. Conversely, larger time windows lead to a higher computational complexity (up to 15 CPU-time seconds for CBC), and a remarkable variability in terms of shape of the generated solution plans.

3.3 Assess the Feasibility of Roadworks

Noteworthy, the proposed approach can also be a valuable tool for assessing the feasibility of performing the required roadworks in the considered period of time, with the required constraints in terms of companies and maximum works per area.

As a case study, we manually designed scenarios where no feasible plan exists. This has been done by providing time windows for roadworks which are smaller than the actual duration of the works, or by forcing too many works to be done in parallel in the same area/by the same company. In the mentioned case studies,

all the solvers very quickly (less than 0.2 CPU-time seconds) identified that no plan could satisfy all the constraints.

Given the ability of the proposed approach that quickly identified infeasible scenarios, it can be used for evaluating the usefulness of additional resources, or for relaxing some of the constraints. Different solutions can be tested and compared. In many of the generated cases, the best solution was to re-negotiate initial time or deadlines for roadworks; in some cases our analysis indicated that the only viable solution was to relax some of the constraints.

Unfortunately, the considered solvers are not designed to indicate which of the constraints is the most stringent, or which resources are the most limiting. In that, the evaluation and exploration process has to be performed manually, by investigating how relaxing some constraints or increasing some resources affect the solvability of the problem at hand.

4 Discussion

The experiments demonstrate the extent to which the proposed approach is able to efficiently and effectively generate roadwork plans according to specified constraints and available resources. Given the notion of quality, that in our case corresponds to starting works as early as possible, MIP solvers can generate optimised solutions for the problem at hand. Remarkably, the quality metric can be easily modified using the MiniZinc language, and can therefore be specified by the local authorities on the basis of their requirements. Different quality metrics can also be specified and mixed together, for instance using a weighted sum of the relevant aspects to be considered. In cases where a quality metric can not be easily specified, a very intuitive notion of quality can be included in the MIP model, and the generated best solutions –all the solutions of the best quality found can be returned by the solvers– can then be manually compared.

The proposed model includes a number of constraints, focusing on the area of the region and on the company performing the works. It should be noted that the set of constraints can be easily extended. For instance, the number of roadworks performed per time unit in the whole controlled region can be limited, or restrictions can be specified in terms of works performed on the same road of the urban network, or on directly connected roads. It is also possible to provide constraints that vary over time: for instance the maximum number of works that a company can perform in parallel may be affected by other commitments of the company in the modelled period, that should be taken into account. Constraints can also be specified in terms of minimum required traffic throughputs for roads or areas. From this perspective, the model can encode a notion of usual throughput of roads and expected reduction due to a given roadwork: the planning engine can take these elements into account in order to generate plans where a minimum throughput is guaranteed.

In the performed experiments, we focused on cases where no works were already planned for the controlled region during the modelled time period. However, roadworks that are already planned, or are already being performed can

be included in the problem model. This can be done in two main ways: by introducing works which have a time window of the same size of the duration, and that corresponds to the period of time were works are being executed; or by implicitly modelling the work by increasing the number of works performed in the relevant area, by the relevant company, in the affected weeks. In the former case, the solver will have to take into account the work explicitly, as if it has yet to be planned. In the latter case, the work is modelled only in terms of its impact on the constraints and resources, so the solver does not need to take it into account when solving the problem.

When comparing the approach introduced in this paper with the preliminary work on using AI Planning for dealing with roadworks scheduling [14, 8], it was possible to make a number of observations. In terms of quality of the generated schedules, measured as the average delay of planned roadworks with regards to the earliest possible start date, MIP-based approaches tend to provide consistently good solutions; Planning-based techniques instead show a very high variability, making virtually impossible to estimate whether using the approach will be beneficial or not. On the other hand, planning engines are generally able to provide solutions very quickly, while MIP-based solvers can be much slower. Given that, it is easy to highlight the high complementarity of the approaches. Planner can be used to quickly provide potentially low-quality scheduling of roadworks, that can be then refined using MIP approaches. Furthermore, schedules generated via Planning can be used as initial solutions for MIPs, that are then tasked to improve them.

Finally, it is worth noting that the proposed approach does not take into account uncertainty of the duration of roadworks. Here we assume that provided expected durations are reasonably accurate, with regards to the considered time unit. In our experimental analysis, given the selected time unit, any discrepancy of less than a week can be ignored. On the other hand, uncertainty can also be taken into account by providing an overestimated duration of the critical roadworks, or by generating problems where works at-risk have in turn different durations: comparing the generated solutions would then shed some light into the most robust scheduling.

5 Conclusion

In order to assist road agencies and authorities in the critical duty of planning essential roadworks, here we investigated the use of Mixed-Integer Programming techniques to provide an effective decision support tool for roadworks planning. Main contributions of this work are: (i) A formalisation of the roadworks planning problem; (ii) A MiniZinc model encoding the problem; and (iii) A thorough experimental analysis, that considers realistic scenarios and compared plans generated using three different MIP solvers, supported by the MiniZinc framework.

The performed experimental analysis demonstrates the extent to which the proposed approach is able to efficiently and effectively create a roadworks plan that satisfies the identified constraints, and follows the provided notion of quality. Remarkably, plans are generated very quickly –in a few seconds–, and this

demonstrates the potential of the approach for being routinely used, also as a feasibility tool, and for assessing different constraints or resources. Furthermore, on the MIP side, we provided class of problems that can be used to benchmark the performance of MIP solvers.

Future work will focus on further validating the proposed approach in different urban regions, and in exploring approaches to explicitly represent uncertainty.

Acknowledgements. L. Chrpa was partially funded by the Czech Science Foundation (project no. 18-07252S). M. Vallati was partially supported by the EPSRC grant EP/R51343X/1 (AI4ME).

References

1. CSPLib language minizinc. `http://www.csplib.org/Languages/MiniZinc`
2. Bixby, E.R., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: Mip: Theory and practice — closing the gap. In: Powell, M.J.D., Scholtes, S. (eds.) System Modelling and Optimization. pp. 19–49 (2000)
3. Burningham, S., Stankevich, N.: Why road maintenance is important and how to get it done (2005)
4. Forrest, J., Lougee-Heimer, R.: Cbc user guide. In: Emerging theory, methods, and applications, pp. 257–277. INFORMS (2005)
5. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., Medina-Carnicer, R.: Generation of fiducial marker dictionaries using mixed integer linear programming. Pattern Recognition **51**, 481 – 491 (2016)
6. Huerne, H.t., Berkum, E.v., Hermelink, W.: A multi objective framework to optimise planning of road works. In: 5th Eurasphalt and Eurobitume Congress (2012)
7. Hussain, R.S., Ruikar, K., Enoch, M.P., Brien, N., Gartside, D.: Process mapping for road works planning and coordination. Built Environment Project and Asset Management **7**(2), 157–172 (2017)
8. M. Ghallab and D. Nau and P. Traverso: Automated Planning Theory and Practice. Elsevier Science (2004)
9. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: Proc. of CP. pp. 529–543 (2007)
10. Paquay, C., Schyns, M., Limbourg, S.: A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. International Transactions in Operational Research **23**(1-2), 187–213 (2016)
11. Schulte, C., Stuckey, P.J.: Efficient constraint propagation engines. ACM Transactions on Programming Languages and Systems (TOPLAS) **31**(1), 2 (2008)
12. Stuckey, P.J., Feydy, T., Schutt, A., Tack, G., Fischer, J.: The minizinc challenge 2008–2013. AI Magazine **35**(2), 55–60 (2014)
13. Valicka, C.G., Garcia, D., Staid, A., Watson, J.P., Hackebeil, G., Rathinam, S., Ntaimo, L.: Mixed-integer programming models for optimal constellation scheduling given cloud cover uncertainty. European Journal of Operational Research (2018)
14. Vallati, M., Chrpa, L., Kitchin, D.: How to plan roadworks in urban regions? a principled approach based on ai planning. In: Proc. of ICCS (2019)
15. Ye, Y., Grossmann, I.E., Pinto, J.M.: Mixed-integer nonlinear programming models for optimal design of reliable chemical plants. Computers & Chemical Engineering **116**, 3 – 16 (2018)