EPiC Series in Computing

Volume 50, 2017, Pages 93–107

GCAI 2017. 3rd Global Conference on Artificial Intelligence

A Genetic Algorithm for Truck Dispatching in Mining

Wesley Cox[1], Tim French[1], Mark Reynolds[1], and Lyndon While[1]

Computer Science and Software Engineering, The University of Western Australia, Perth, Australia
{wesley.cox,tim.french,mark.reynolds,lyndon.while}@uwa.edu.au

Abstract

We apply genetic algorithms (GAs) to evolve cyclic finite automata for scheduling the dispatch of trucks in mines. The GA performs well generally, and on problems which include one-lane roads, the GA was able to find solutions that utilised shovels very well, with low contention and using fewer trucks than both the widely-used linear programming DISPATCH algorithm, and commonly-used greedy heuristics. The GA provides significant cost-savings, or production increases, on problems where alternative algorithms do not adapt well.

1 Introduction

Industrial supply chain management and scheduling tasks are a serious real-world application of temporal reasoning techniques. Traditional techniques are often based on linear programming (LP), specific heuristics or queuing theory. In this paper we aim to consider whether evolutionary search techniques coupled with timed automata can compete with traditional methods.

In mines, trucks travel between shovels, which load trucks with ore, and crushers, where trucks unload ore. When a truck has finished being serviced at a location it requires a new destination, provided by the dispatcher. Variable conditions in the mine necessitate real-time decision making. Poor dispatching choices can lead to long queues at some shovels, while others go under-utilised. To solve this problem, automated dispatching methods can be used to optimise the throughput of a mine. Truck haulage typically represents 50-60% of mining costs [1], so a dispatching method should minimise the number of trucks required to achieve good throughput. Relative small increases in production can be highly profitable in this industry.

We apply genetic algorithms (GAs) to produce a controller for dispatching decisions. GAs are a stochastic search technique inspired by natural selection [26]. Solutions are represented as chromosomes that reproduce through crossover and mutation, and are evaluated by a fitness function. A population of high performing solutions survives and reproduces, leading to even better solutions. We use a GA to evolve sets of cyclic finite automata that are used to perform dispatching decisions. Dispatching locations each cycle through a list of destinations, which have been optimised by the GA for the particular mine.

A common industry approach used in over 200 mines is the DISPATCH software developed by Modular Mining Systems [13]. The dispatching algorithm used by DISPATCH was originally presented by White and Olsen [24] and White *et al.* [25], though it has been assumed that due to its commercial status, not all details were provided [1]. The DISPATCH commercial software is a fully fledged mine management system and it seems likely that the dispatching algorithm itself would have evolved since the original theoretical algorithm was presented. We refer to the original dispatching algorithm simply as DISPATCH. Using a discrete event simulator, our approach is compared against common greedy heuristics, as well as an adaptation of DISPATCH. The heuristics and DISPATCH are shown to be insufficient on complicated set-ups.

We test the approach on three different problem types of varying complexity. The first two use only two-lane roads and previous algorithms are sufficient. The third problem introduces one-lane roads, requiring a more global solution to handle contention in the road network. The GA is shown to achieve high performance on the third problem with fewer trucks than alternative algorithms.

The rest of the paper is structured as follows. Section 2 describes previous work on real-time dispatching strategies in open-pit mines. Section 3 describes our approach to dispatching and the model used for testing. Section 4 describes the experiments and discusses the results. Section 5 provides the conclusion.

2 Previous Work

Dispatching strategies in open-pit mines have been summarised by Munirathinam and Yingling [14] and Alarie and Gamache [1]. Presented here are brief summaries.

Many greedy heuristics have been summarised by Tan and Ramani [22]. Common greedy heuristics are: *Minimise Truck Cycle Time (MTCT)*, dispatch to the shovel from which the truck is expected to return soonest; *Minimise Truck Waiting Time (MTWT)*, dispatch to the shovel where the truck is expected to wait the least time; *Minimise Truck Service Time (MTST)*, dispatch to the shovel where the truck is expected to be serviced in the least time; and *Minimise Shovel Waiting Time (MSWT)*, dispatch to the shovel that has been waiting longest, or will be available soonest.

Several dispatching strategies of varying complexity have been described in the literature. The less greedy approaches are commonly plan-driven, i.e. real-time dispatching is performed based on a predetermined operational plan; and produce temporary schedules, i.e. each decision considers multiple trucks but only the most urgent assignment is kept.

Hauck [10] presents an approach where scheduling is performed by solving a series of assignment problems based on integer equations that consider the actions of each truck at discrete time steps, and recent average travel and loading times. The system attempts in real-time to minimise shovel idle time and time spent with trucks travelling or waiting.

Soumis *et al.* [17, 16] produce an operational plan by solving a non-linear program that considers blending requirements, waiting times derived from queuing theory, and availability of trucks. Dispatching produces a temporary schedule by solving an assignment problem that minimises deviation from the plan, and considers probability distributions of expected waiting times at each shovel. No mathematical formulations are presented.

White and Olsen [24] and White *et al.* [25] present DISPATCH, a plan-driven approach. In the offline stage, an operational plan, specifying the desired haulage rates along available routes, is produced by solving two LPs which consider shovel and crusher rates, grade requirements and network constraints. In the online stage, routing uses temporary schedules, created by ordering shovels by a neediness function that considers haulage rates and travel times, and assigning trucks to minimise a lost-tons function that considers truck and shovel idle times.

Li [11] produces an operational plan that optimises truck flow along available paths by solving an LP based on travel times and predetermined ore requirements. Dispatching greedily chooses the shovel with the highest ratio between the time since its last dispatch and the optimal inter-arrival times determined by the LP. It ignores the current state of the mine, thus it does not compensate for potential queues.

Bonates [7] presents some variations on the common greedy heuristics.

Temeng *et al.* [23] produce an operational plan that optimises production rates along available routes by solving a goal-programming model. Dispatching produces a temporary schedule that attempts to minimise total waiting time while considering shovel demand, based on how far behind schedule each shovel is.

Bissiri [6] applies an agent-based system inspired by a social insect model. Shovels bid for trucks, which are reassigned if demand exceeds a threshold value based on its capacity, remaining distance to its current assignment, its previous assignment, and current road conditions.

Ta [21] and Ta *et al.* [19] apply stochastic programming to produce a fixed truck assignment. Their model minimises the resources required to satisfy ore demand subject to operating constraints, to a specified probability.

Bastos [4] and Bastos *et al.* [3] apply Time-Dependent Markov Decision Processes (TiMDP) to model the stochastic behaviour of a mine. A single-dependent-agent approach is applied to reduce the size of the solution space and solve for a dispatching policy on the reduced TiMDP. In [4] the TiMDP is combined with a GA.

Ta *et al.* [20] uses queueing theory to produce a fixed truck assignment. They solve an LP which minimises the number of trucks required to achieve a given total shovel throughput subject to blending constraints, where throughput is estimated using queueing theory concepts.

Subtil *et al.* [18] describe a brute-force approach used in SmartMine that runs an exhaustive search over upcoming decisions. The size of the search space is not specified.

Underground mines add further complexity and constraints including one way roads, reduced communication between vehicles and limited awareness of vehicle locations [15] making traditional scheduling techniques unviable.

3 Approach

3.1 Simulation Model

To compare dispatching algorithms, a simulator was designed around a model of a mine site based on a network of timed automata (TA). TA are finite state automata extended with a set of real-valued clocks [2]. Transitions between states can be dependent on the values of these clocks. A simplified example of a TA representing a truck is shown in Figure 1. In the simple case the truck moves through the mine in a *travel-queue-fill-travel-queue-empty* cycle. In the model, when a truck enters a timed state (i.e. travelling, filling, emptying), the total time it will spend in that state is restricted to a fixed range. In the simulator, upon entering these states the transition time to the next state is randomly generated from a uniform distribution. This is used to model the variations that occur in a real-world system.

The simulator is event-based, where each event represents a change in state for either a truck or a traffic light, and the current shift-time in the simulation is advanced accordingly. Additionally, for any point in time, each truck can be assigned an effective position in the mine represented as a state-value pair. When filling or emptying, the value reflects the portion of the service that has been completed; when travelling, the value reflects the position of the truck on a road, calculated by assuming an initial constant speed then considering potential slowdowns if catching up to slower trucks on that road, as overtaking is not allowed. These state-value pairs alone cannot be used to determine future transition times, but can be used to initialise a separate simulation for estimating the heuristic values used by the greedy algorithms.

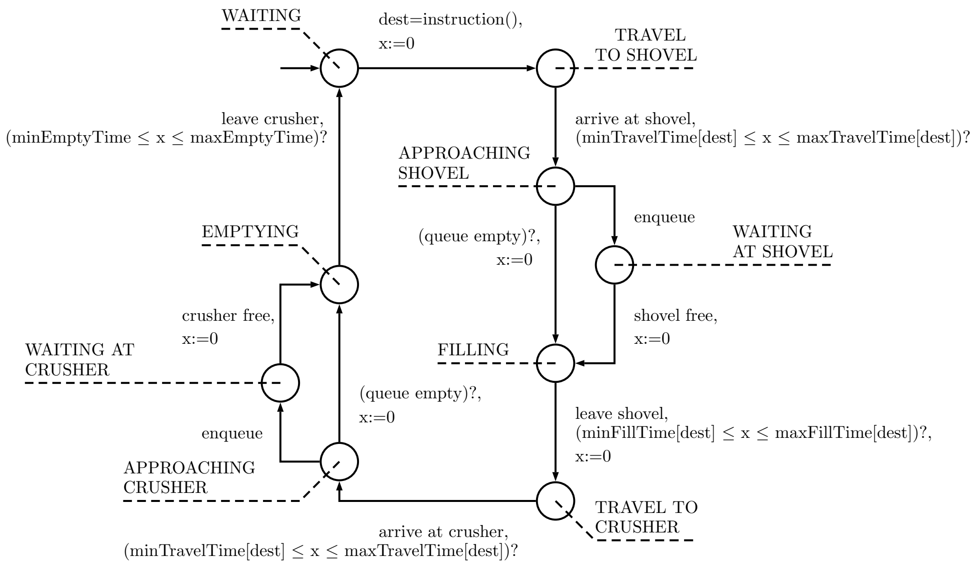


Figure 1: A simplified example of a TA model of a mine truck, with one clock x. Certain transitions can only be taken when x lies within a specific range. Not included in this example are traffic lights, routes comprising multiple roads, and multiple crushers.

3.2 Genetic Algorithm

We applied a GA to evolve a controller for making dispatching decisions. The controller is a set of cyclic finite automata, each equivalent to a list of destinations which the controller cycles through. Each dispatching location (where more than one outgoing destination is possible) has its own cycle. The GA optimises the controller for a given mine instance and a fixed number of trucks. This is an offline algorithm; controllers are produced for entire shifts by running the GA beforehand, rather than making real-time short-term decisions.

Constructing a GA requires a definition of the chromosome genotype, specifically how solutions are represented; a fitness function for evaluating chromosomes; genetic operators for producing new chromosomes from existing ones, such as crossover and mutation; and a selection scheme, for determining which chromosomes survive and reproduce based on their fitness.

In our GA, a chromosome is represented by n variable-length integer strings, for n dispatching locations. For example, consider a mine with 2 crushers and 3 shovels. Each chromosome would have 5 integer strings: one possibility would be $[[0,0,0,1],[0,1,2],[0],[1],[0,1]]$. Here the first crusher repeats the pattern $[0,0,0,1]$, dispatching to the first shovel 3 times, then to the second shovel once. This occurs independently of the second crusher, which repeats the pattern $[0,1,2]$, dispatching once to each shovel. The last 3 strings are used by the shovels. The first 2 each exclusively dispatch to 1 crusher, while the third alternates between the crushers.

The fitness function uses the simulator to run a stochastic simulation of the mine, given a fixed number of trucks and using the provided controller for dispatching decisions, and outputs the number of truckloads emptied at the crushers in a shift. Because of the noisy fitness function, the fitness of a chromosome is the average of multiple fitness evaluations. Evaluations are stored in a bucket and a new evaluation is performed for each surviving chromosome in each generation, with the oldest evaluation discarded from the bucket.

Crossover combines two chromosomes by crossing each pair of strings independently. Pairs of strings are crossed by randomly choosing a single crossover point on each string then swapping values past those points. The portion taken from each string can be of any length, producing child strings with different lengths to their parents. Four different forms of mutation can occur. *Value mutation* replaces one element in one string with a random value; *Insertion* inserts a random value into one string; *Deletion* deletes one element from one string; *Inversion* reverses a random portion of one string.

Reproduction selection is uniform from the population. Survival selection is partially by elitism, and the remaining selection is by tournament selection with a tournament size of 4, with candidates chosen from the pool of both parents and offspring. Tournament selection selects a candidate by comparing t random members of the selection pool, for tournament size t, and returning the candidate with the best fitness [12].

Parameters for the GA were chosen based on early experimentation. The crossover rate was set to 0.9, and each form of mutation occurs independently with a rate of 0.05. The elitism rate was set to 0.1. Each run of the GA was 500 generations. 100 chromosomes survive each generation, and 200 new offspring are produced each generation. The fitness bucket size is 20.

4 Results and Discussion

4.1 Experimental Methodology

Testing was performed in a discrete event simulator, as described in Section 3.1, implemented in Java. Code used for this paper is publicly available [8]. All trucks have the same speed distribution and each truckload is the same size. Crushers and shovels are heterogeneous; their average servicing rates vary from machine to machine. All trucks start the shift empty at a crusher. Full trucks have 20% greater travel times than empty trucks. Overtaking is banned.

Three types of problems were considered, with different amounts of equipment and different road types joining them. The basis for these different problem types is simply to test the ability of each algorithm to adapt to added complexities. Examples of the three road networks are shown in Figure 2.

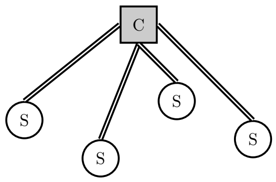
Problem 1 has 4 shovels and 1 crusher, and each shovel has one distinct two-lane road to the crusher. *Problem 2* has 6 shovels and 2 crushers, and each shovel-crusher pair is joined by one distinct two-lane road. *Problem 3* has 6 shovels and 2 crushers, and each shovel-crusher pair is joined by one distinct route with a two-lane road connected to the crusher joined to a one-lane road connected to the shovel. One-lane roads can be traversed in both directions, managed by traffic lights that respond to arrivals; a green light means that no trucks are travelling or waiting in the opposite direction.

In Problems 2 and 3, the crushers are not next to each other; shovels close to one crusher are far from the other and within each instance the average travel time between crushers is similar across routes (ignoring service and waiting times).

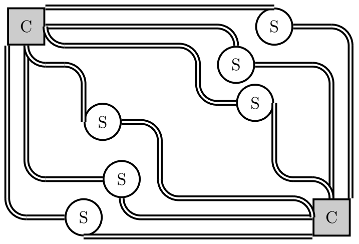
Six instances of each problem were randomly generated, with different service rates for each machine, different travel times for each route, and in Problem 3 different two-lane to one-lane ratios for each route. The average emptying rate was set to 1/3 (trucks per minute) and the total filling rate for all shovels is equal to the total emptying rate for all crushers.

4.2 Previous Algorithms

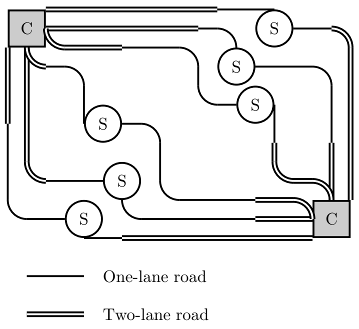
The GA approach was compared with the greedy heuristics MTCT, MTWT, MTST, and MSWT, as well as DISPATCH [24, 25], due to its status in the industry. Each of these previous



(a) Problem 1. 1 crusher (C) and 4 shovels (S).



(b) Problem 2. 2 crushers (C) and 6 shovels (S).



(c) Problem 3. 2 crushers (C) and 6 shovels (S). Each route consists of a two-lane road connected to a crusher that narrows to a one-lane road connected to a shovel.

Figure 2: Example road networks for each test problem.

algorithms are online algorithms. Other algorithms considered from the literature either performed poorly in early experiments, were unsuitable for our problem specifications, or lacked sufficient detail for a suitable implementation. Each algorithm used in testing was implemented in Java by the authors. LPs were solved with the *lpsolve* library [5]. Some adaptations were necessary to fit the algorithms to our problem specification, which are described below.

4.2.1 Greedy Heuristics

We implemented the greedy algorithms to calculate their heuristic values by performing 20 stochastic forward simulations, considering the current locations of all trucks in the mine. In multi-crusher problems, a pair of dispatches is produced; the first is the route to a shovel which is used immediately, and the second is the return route which is stored. MTCT evaluates each pair of possible routes. MTWT, MTST, and MSWT first find the route that best satisfies their heuristic, then given that choice find the return route that best satisfies that heuristic (e.g. MTST finds the shovel that will service the truck soonest, then given that assignment finds the crusher that will service that truck soonest). The concept of a shovel being available soonest makes less sense in a multi-crusher problem, so in Problems 2 and 3 MSWT finds the shovel with the expected earliest time window when a truck can be serviced.

4.2.2 DISPATCH

DISPATCH relies on linear programming to determine the desired flow for each route. The original LPs described attempt to minimise costs and trucks while satisfying production demand, and feature constraints not relevant to our problem specification such as blending constraints. We have used a similar LP, detailed in Appendix A, which discards unused constraints and attempts to maximise production given a preset fixed number of trucks.

This approach has issues on Problem 3 however, because both the LP and dispatcher require an estimate of the expected time to traverse each route. The expected time to clear a one-lane road includes the expected waiting time at traffic lights, which depends on the flow in both directions. Through simulations it was observed that, if the flow F in both directions on a one-lane road is equal, then the expected waiting time is equal to the mean travel time T if $F > \frac{1}{2T}$, otherwise it can be approximated as $\frac{FT^2}{2}$ (except when F is very close to $\frac{1}{2T}$). To adapt DISPATCH to Problem 3, the LP and dispatcher estimate the expected waiting time of one-lane roads using this approximation where F is set as the filling rate of the shovel that road feeds. This is based on the assumption that most of the flow to and from a shovel is from a single crusher, based on observing this on Problem 2, and thus the assigned flow on any route to a shovel is expected to be usually either 0 or the filling rate, and equal in both directions.

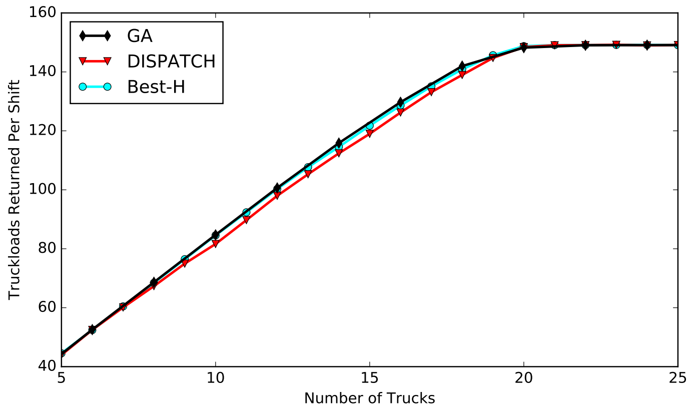
4.3 Results

The dispatching algorithms were compared on multiple instances of each problem. Figures 3-5 show for typical instances the average production of a 500 minute shift, measured in complete truckloads returned to the crushers by shift end, against the number of trucks available. Each data point is the average of 50 shifts. Results for the heuristics show an algorithm portfolio [9] of the 4 heuristics, showing the best performing heuristic at each given number of trucks, henceforth referred to as Best-H. Results for the GA show the 90th percentile of 20 runs (i.e. the 3rd best run) for each number of trucks; this percentile is used in further discussion. Due to their stochastic nature, it is common to run GAs multiple times to get better results, which is quite practical since the GA is run offline. The 90th percentile was chosen, as opposed to the absolute best solutions, to demonstrate reliability. Standard deviations for each line are typically small (often less than 1, usually less than 2 truckloads), and thus are not shown.

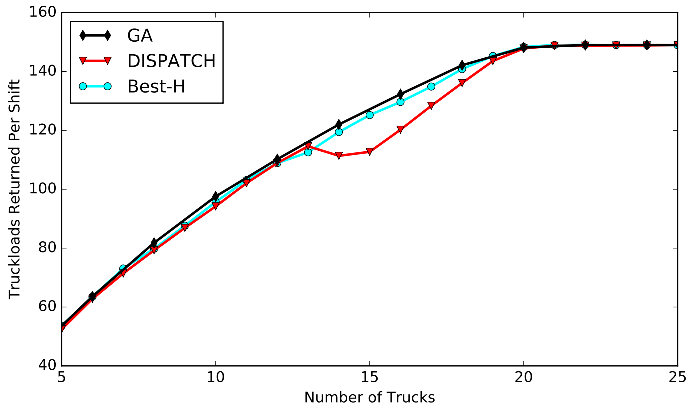
4.3.1 Problem 1

Performance on Problem 1 is shown in Figure 3. The GA easily performs as well as the best-performing algorithm for any number of trucks, consistently converging to a solution of the same quality. The low complexity means the problems are easily solved, even with noise.

DISPATCH performs well, except in instances where one shovel is much farther from the crusher than the others and the system is undertrucked. In these cases, DISPATCH favours the distant shovel too much and performance suffers, as observed in Figure 3b. The original algorithm featured a step to decide how many trucks to use, thus the real-time dispatcher is not designed to operate on undertrucked systems, even if the flow is adjusted to compensate for having fewer trucks available as it is here. However, DISPATCH still reaches its optimal production level in the same number of trucks as the GA.



(a) Instance 2



(b) Instance 6

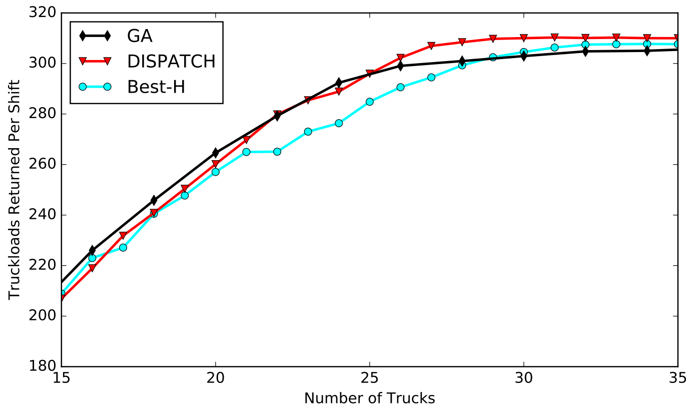
Figure 3: Comparison of dispatching algorithms on typical instances of Problem 1. Plotted is the average number of truckloads returned to the crusher per shift against the number of trucks available.

On typical instances, MTWT reaches the optimal production level with the fewest trucks, compared to the other heuristics, but performs poorly on undertrucked systems. Best-H reaches optimal production with the same number of trucks as the GA, but for undertrucked systems is outperformed by as much as 4%.

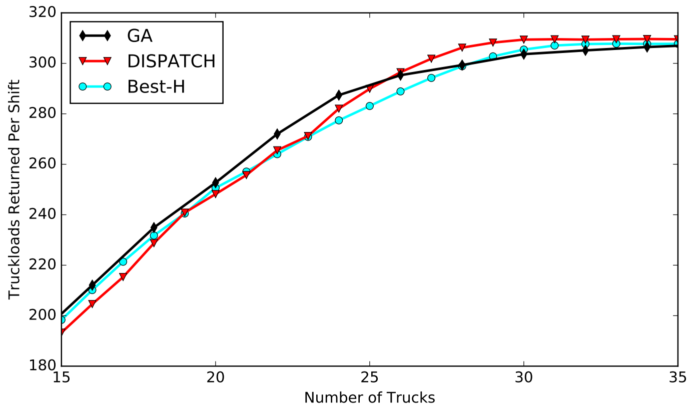
4.3.2 Problem 2

Performance on Problem 2 is shown in Figure 4. The GA is slightly outperformed (by as much as 3%) for saturated networks by DISPATCH. This indicates that for this kind of problem, simple cyclic automata that do not respond to disruptions from noise are insufficient.

The relative order of the performance of the other algorithms differs from Problem 1. MTST now typically reaches the optimal production level with the fewest trucks, compared to the other heuristics. DISPATCH still performs as well as or better than Best-H on saturated systems.



(a) Instance 3



(b) Instance 5

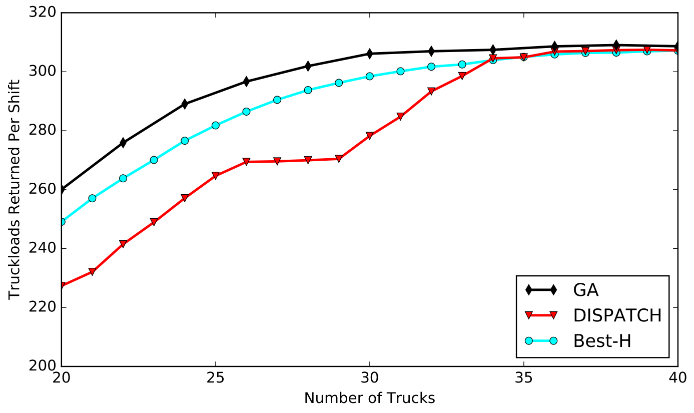
Figure 4: Comparison of dispatching algorithms on typical instances of Problem 2. Plotted is the average number of truckloads returned to the crushers per shift against the number of trucks available.

4.3.3 Problem 3

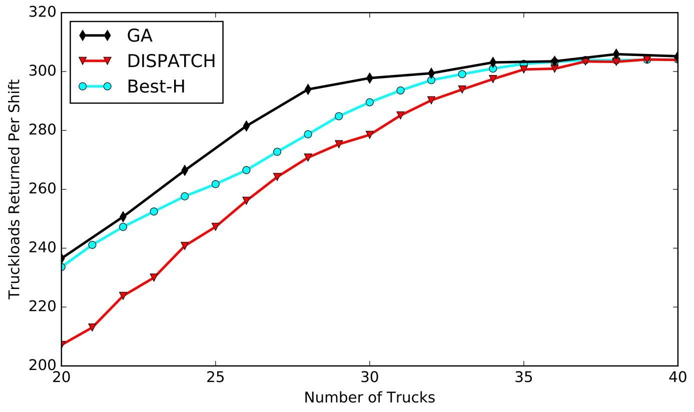
Performance on Problem 3 is shown in Figure 5. Table 1 shows the minimum number of trucks required for Best-H, DISPATCH, and the GA to achieve over 95%, 98%, and 99% of the optimal performance observed on each instance of Problem 3. The GA achieves high performance with fewer trucks than the other algorithms, except on Instance 6, indicating it is suitable as a cost-saving measure. In most cases, the heuristics cannot achieve above 99% of the peak performance with any number of trucks.

The best performing heuristic, for reasonable numbers of trucks, varies between MTST and MTWT. While MTST already performed well on Problem 2, MTWT likely performs well in cases where the need to minimise contention on one-lane roads outweighs considerations of travel times.

DISPATCH performs poorly when the system is undertrucked. This is likely due to the fact



(a) Instance 1



(b) Instance 2

Figure 5: Comparison of dispatching algorithms on typical instances of Problem 3. Plotted is the average number of truckloads returned to the crushers per shift against the number of trucks available.

Table 1: The minimum number of trucks required to achieve at least 95%, 98%, and 99% of the peak performance on 6 instances of Problem 3. Compared are Best-H (B-H), DISPATCH (D), and the 90th percentile for the GA (GA-90). Cells marked as unl. indicate the algorithm cannot achieve this performance with any number of trucks.

	95%			98%			99%		
Instance	B-H	D	GA-90	B-H	D	GA-90	B-H	D	GA-90
1	29	33	26	34	34	29	39	37	32
2	31	33	28	35	37	34	unl.	43	38
3	32	33	30	36	37	34	unl.	39	38
4	33	36	28	38	39	31	unl.	unl.	34
5	41	43	32	48	47	42	unl.	unl.	45
6	27	26	24	30	28	28	32	29	31

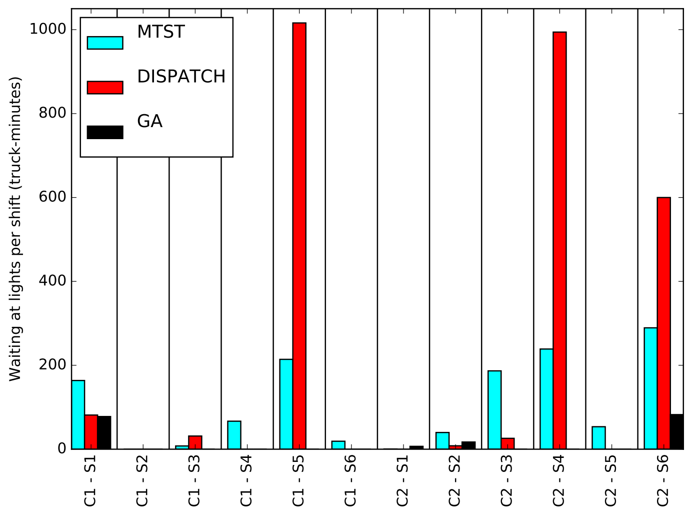


Figure 6: Comparison of productivity loss between MTST, DISPATCH, and the best GA solution (of 20 runs) on Instance 4 of Problem 3 with 31 trucks. Productivity loss is measured in truck-minutes spent waiting at traffic lights for each one-lane road in both directions, averaged over multiple shifts. Ci-Sj refers to the route between crusher i and shovel j.

that in these cases some shovels are under-utilised, thus the expected waiting times on one-lane roads are less accurate.

4.3.4 Discussion

Figure 6 demonstrates the effects of contention on the one-lane roads, by showing the average truck-minutes spent waiting at each one-lane road per shift. MTST, DISPATCH, and the best performing solution of 20 runs of the GA are compared here on instance 4 of Problem 3. The number of trucks was set to 31, sufficient for the GA to achieve over 98% of peak performance as shown in Table 1. On the same instance with the same solutions, Figure 7 compares the utilisation of each machine, by showing the percentage difference between the arrival rate at each machine and the service rates, for the final three quarters of the shift. Values close to zero indicate high utilisation of that machine.

The GA manages to match the service rates within 5% for 5 shovels and both crushers. MTST however, only achieves within 5% on 3 shovels and 1 crusher. This shows that for this number of trucks, MTST can only utilise some of the machines well. MTST demonstrates a much higher total loss in truck use due to contention than the GA. Other instances are similar, with MTST showing higher total loss in truck use due to contention, and at least one shovel and crusher showing poor utilisation. MTWT is relatively better than MTST at avoiding waiting times on one-lane roads, but still shows a higher total loss in truck use than the GA.

On all instances, DISPATCH is observed to have significant productivity loss on at least 2 roads, and shovels that are significantly underutilitised are connected to roads with minimal productivity loss. This is attributed to an issue also observed in Problem 1, which is that the dispatching algorithm favours much longer routes, and the underused roads in this case are also the shortest routes. This issue is amplified by the approximation of waiting times on one-lane roads, which is much higher on the longer roads. Thus DISPATCH tends to require more trucks

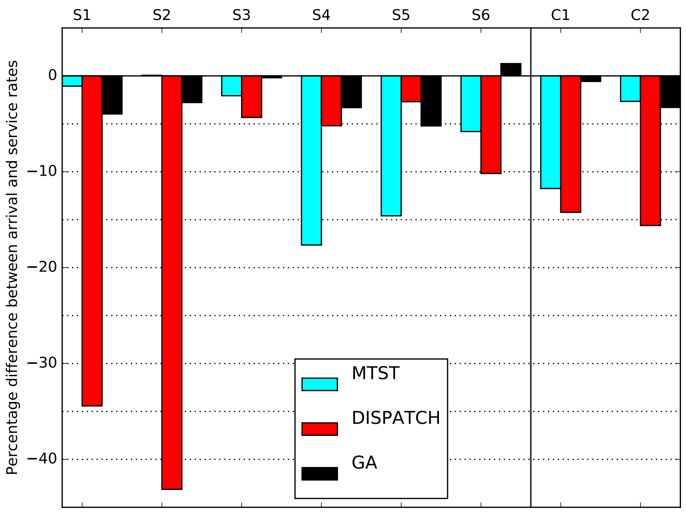


Figure 7: Comparison of shovel and crusher utilisation between MTST, DISPATCH, and the best GA solution (of 20 runs) on Instance 4 of Problem 3 with 31 trucks. Utilisation is measured by the percentage difference between the arrival and service rates, after the first shift-quarter, averaged over multiple shifts.

to satisfy a mine on Problem 3.

The greedy algorithms cannot coordinate trucks in the long-term to minimise contention, even if they otherwise match the correct dispatching rates and favour shorter routes where appropriate. The GA is able to find a balanced solution that can match the required dispatching rates, while managing routes to minimise contention and travel times. On Problem 3, a global solution is appropriate to manage contention, and a GA is a suitable approach to find it.

Because GAs operate with a black-box fitness function, they can solve a wide range of problems. In this case, the GA-based approach is not limited to these specific problem types. The simulation model, used for the fitness function, can be adapted to many different road networks and problem configurations. The compared algorithms however, use heuristic values which evidently do not adapt well to more complex problems.

5 Conclusion

We have used a genetic algorithm to evolve cyclic automata for dispatching trucks in mines. Solutions found by the GA were compared with common greedy heuristics, as well as the DISPATCH algorithm, on three problem types of varying complexity. On Problem 1, with only one crusher, the GA easily constructed solutions that performed as well as the alternatives. On Problem 2, with two crushers, the GA sometimes performed slightly worse than the alternatives due to the inability of the cyclic automata to react to disruptions. On Problem 3, which introduced one-lane roads, the GA was able to achieve high performance with fewer trucks than the alternatives. The benefits it gained from producing a more global solution that reduced contention outweighed issues caused by disruptions.

Future work will focus on real-time evolution of schedules, and on generalising the problem to include blending constraints and issues specific to underground mines.

References

[1] Stéphane Alarie and Michel Gamache. Overview of solution strategies used in truck dispatching systems for open pit mines. *International Journal of Surface Mining, Reclamation and Environment*, 16(1):59–76, 2002.

[2] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[3] Guilherme S. Bastos, Luiz E. Souza, Fabio T. Ramos, and Carlos H. C. Ribeiro. A single-dependent agent approach for stochastic time-dependent truck dispatching in open-pit mining. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1057–1062. IEEE, 2011.

[4] Guilherme Sousa Bastos. *Methods for Truck Dispatching in Open-pit Mining*. PhD thesis, Aeronautics Institute of Technology, São José dos Campos, 2010.

[5] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. lpsolve. `https://sourceforge.net/projects/lpsolve/`, 2006–2016.

[6] Yassiah Bissiri. *Application of agent-based modeling to truck-shovel dispatching systems in open pit mines*. PhD thesis, University of British Columbia, 2002.

[7] Eduardo Jorge Lira Bonates. *The development of assignment procedures for semi-automated truck-/shovel systems*. PhD thesis, McGill University, 1992.

[8] Wesley Cox, Lyndon While, Tim French, and Mark Reynolds. GA-TA truck dispatching. `https://github.com/wesleycox/GA-TA-Truck-Dispatching`, 2017.

[9] Carla P Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43–62, 2001.

[10] Robert F. Hauck. Computer-controlled truck dispatching in open-pit mines. In *Computer methods for the 80's in the mineral industry*, pages 739–742. AIME Calgary, 1979.

[11] Z. Li. A methodology for the optimum control of shovel and truck operations in open-pit mining. *Mining Science and Technology*, 10(3):337–340, 1990.

[12] Brad L Miller and David E Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[13] Modular Mining Systems. Dispatch fleet management. `http://www.modularmining.com/product/dispatch/`, 2017. Accessed: 2017-01-13.

[14] Mohan Munirathinam and Jon C. Yingling. A review of computer-based truck dispatching strategies for surface mining operations. *International Journal of Surface Mining, Reclamation and Environment*, 8(1):1–15, 1994.

[15] Zhen Song, Håkan Schunnesson, Mikael Rinne, and John Sturgul. Intelligent scheduling for underground mobile mining equipment. *PloS one*, 10(6):e0131003, 2015.

[16] F. Soumis, J. Ethier, and J. Elbrond. Evaluation of the new truck dispatching in the mount wright mine. *Application of Computers and Operations Research in the Mineral Industry*, pages 674–682, 1989.

[17] F. Soumis, J. Ethier, D. McInnis, and J. Elbrond. *A new method for automatic truck dispatching in an open pit mine*. Ecole Polytechnique de Montreal, 1986.

[18] Robert F. Subtil, Diego M. Silva, and Julio Cesar Alves. A practical approach to truck dispatch for open pit mines. In *35th APCOM symposium*, pages 24–30, 2011.

[19] C. H. Ta, J. V. Kresta, J. F. Forbes, and H. J. Marquez. A stochastic optimization approach to mine truck allocation. *International journal of surface mining, reclamation and environment*, 19(3):162–175, 2005.

[20] Chung H. Ta, Armann Ingolfsson, and John Doucette. Haul truck allocation via queueing theory. *European Journal of Operational Research*, 231(3):770–778, 2010.

[21] Chung Huu Ta. *Optimal haul truck allocation in the syncrude mine*. PhD thesis, University of

Alberta, 2002.

[22] Sizhe Tan and R. V. Ramani. Evaluation of computer truck dispatching criteria. In *Proceedings of the SME/AIME annual meeting and exhibition, Arizona*, pages 192–215, 1992.

[23] Victor A. Temeng, Francis O. Otuonye, and James O. Frendewey Jr. Real-time truck dispatching using a transportation algorithm. *International Journal of Surface Mining, Reclamation and Environment*, 11(4):203–207, 1997.

[24] J. W. White and J. P. Olson. Computer-based dispatching in mines with concurrent operating objectives. *Min. Eng.(Littleton, Colo.);(United States)*, 38(11), 1986.

[25] J. Wm. White, J. P. Olson, and S. I. Vohnout. On improving truck/shovel productivity in open pit mines. *CIM bulletin*, 86:43–43, 1993.

[26] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

A Adapted LP for DISPATCH

$$\max \sum_{i=1}^{Nc} MC_i - T$$

$$\text{s.t. } C_i \le Cr_i,\ 1 \le i \le Nc$$

$$C_i = \sum_{j=1}^{Ns} \overrightarrow{F}_{i,j} = \sum_{j=1}^{Ns} \overleftarrow{F}_{i,j},\ 1 \le i \le Nc$$

$$S_j \le Sr_j,\ 1 \le j \le Ns$$

$$S_j = \sum_{i=1}^{Nc} \overrightarrow{F}_{i,j} = \sum_{i=1}^{Nc} \overleftarrow{F}_{i,j},\ 1 \le j \le Ns$$

$$T = \sum_{i=1}^{Nc} \frac{C_i}{Cr_i} + \sum_{j=1}^{Ns} \frac{S_j}{Sr_j} + \sum_{i=1}^{Nc}\sum_{j=1}^{Ns}(R_{i,j}\overrightarrow{F}_{i,j} + R_{i,j}Fp\overleftarrow{F}_{i,j})$$

$$T \le Nt$$

for (C_i, S_j) total truck flow through (crusher i, shovel j) (trucks/min) [to be determined]

(Cr_i, Sr_j) (crusher i, shovel j) service rate (trucks/min) [specified]

$\overrightarrow{F}_{i,j}$ truck flow from crusher i to shovel j (trucks/min) [to be determined]

$\overleftarrow{F}_{i,j}$ truck flow from shovel j to crusher i (trucks/min) [to be determined]

$R_{i,j}$ expected route clear time between crusher i and shovel j (min) [specified]

Fp full truck speed penalty [specified]

(Nc, Ns) number of (crushers, shovels) [specified]

Nt maximum number of trucks available [specified]

T estimated number of trucks to satisfy total system flow [to be determined]

M large constant