MASTER'S THESIS 2019

Improving Real-time Scheduling by Evolutionary Algorithms

Hedda Malmström, Sofia Tenerz

ISSN 1650-2884
LU-CS-EX 2019-11

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2019-11

Improving Real-time Scheduling by Evolutionary Algorithms

Hedda Malmström, Sofia Tenerz

Improving Real-time Scheduling by Evolutionary Algorithms

(Utilising Information from Previous Schedules to Enhance Performance)

Hedda Malmström
ine14hma@student.lu.se

Sofia Tenerz
ine14ste@student.lu.se

May 31, 2019

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Elin Anna Topp, elin_anna.topp@cs.lth.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Building on previous studies investigating the use of real-time evolutionary algorithms for truck dispatching scheduling in underground mines, this study aims to investigate the potential improvement in performance that can be achieved by using information from previously used and discarded schedules when constructing new ones. Changes to two different algorithms have been made; a single evolutionary algorithm developing truck dispatching schedules and a co-evolutionary algorithm developing truck dispatching schedules and traffic light schedules in parallel. For both respectively, two different approaches have been investigated.

The changes to the evolutionary algorithm result in similar mine productivity compared to the original approach, but a decrease in the required computational power. For the co-evolutionary algorithm, the changes have led to a large decrease in computational power, which comes at a cost of a slight decrease in productivity.

Keywords: Machine Learning, Evolutionary Algorithms, Vehicle Dispatching Problem, Underground Mine Dispatching, Real-time Scheduling

Acknowledgements

First of all we would like to thank Lyndon While and Wesley Cox at the Department of Computer Science and Software Engineering at University of Western Australia. Thank you for having us and for supporting us along the way with both suggestions on the overall focus of the study, and with more specific issues and obstacles.

We would also like to thank our supervisor Elin Anna Topp for guiding us and giving us valuable feedback moving forward, no matter where in the world we have been positioned. We have appreciated your help a lot. Marcus Klang at the Department of Computer Science at Lund University has also been an incredible resource after coming back to Sweden.

Lastly, we want to express a huge thank you to both of our families for supporting us throughout these past five years, we could not have done this without you.

Contents

Chapter 1
Introduction

This chapter gives an initial understanding of the purpose of the study, the problem at hand and the research question the study aims to answer as well as the associated limitations. Thereafter, the authors' perceived contributions in relation to one another as well as to academia are described. Lastly, the general outline of the report is explained. Some of the more specific terminology that is used throughout this chapter is listed and explained in more detial in Section 2.1.2.

1.1 Purpose

In underground mines, a challenge arises in regards to scheduling of trucks carrying ore from the point where it is being broken to where it is further processed. The problem this constitutes is very similar to a traditional vehicle dispatching problem but with the addition of features specific to the mine context. This challenge has been addressed through many different means, which will be discussed in Background (Section 2, primarily in 2.2, 2.3. Using evolutionary algorithms (henceforth referred to as EAs) to develop suitable schedules has been found to be one successful approach. At University of Western Australia (henceforth referred to as UWA) much work has been done to evaluate the use of EAs to evolve schedules for real-time underground mine scheduling.

This study builds directly upon the work performed at UWA, and more specifically on their latest research project which was compiled into a submission finalised in late 2018 [7]. The key concepts and findings of that project are described in Previous work (Section 2.3), which makes up the first stepping stone for this study.

The purpose of this study is to further enhance the EA developed by UWA in order to improve its performance in the context of real-time scheduling in underground mines. This will be done by examining the use of already existing information when rerunning the EA, as opposed to the benchmark approach which is constructed so that the EA begins every run in a randomly generated starting point.

1.2 Problem description

The profitable operation in any underground mine is to extract ore and transport it to the surface. In order to do this, machinery is placed on different levels underground and a number of trucks populate the mine, going back and forth throughout the work shift. The costs incurred by truck transportation alone stand for approximately 50-60 percent of total costs [13]. Due to high costs related to site expansion, certain sections of the roads underground are one-lane only, complicating the traffic situation. This results in a complex scheduling problem, one that aims to secure and ideally maximise productivity while at the same time keeping costs at a minimum.

This reasoning has been captured and modeled in a software mine simulator. This project, as well as the previous work in the area performed by UWA, is centered around this mine simulator and thus rather specific problem setup. The outlines of it are explained in this section.

1.2.1 Mine simulator setup

Figure 1.1 shows the setup of the underground mine simulator. Above ground level a crusher is placed adjacent to the opening of the mine. The crusher is where the trucks unload their ore. Hence, it is the destination of all loaded trucks. When the ore is unloaded the truck gets a new destination; which shovel to go to next. The shovels are placed at different depths and this is where the trucks are loaded with ore. The downward sloping road sections are one-lane only and the roads to and from each shovel are two-lane. At the intersection between the main road and each of the shovel access roads there are so called passing points. At each passing point traffic lights regulates the traffic in both directions.

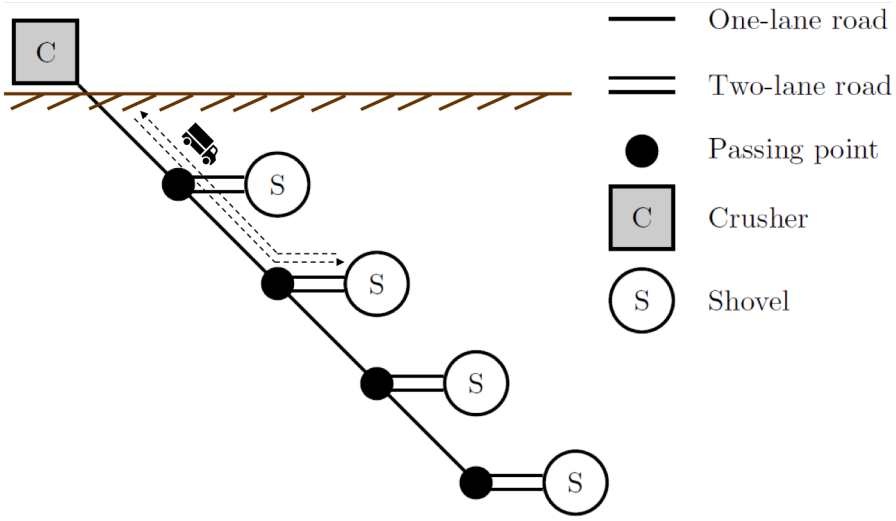


Figure 1.1: Model of an underground mine. A truck is given a destination shovel each time it leaves the crusher. Access to each one-lane road is controlled by a pair of traffic lights.

In order to regulate the traffic in the mine, two separate schedules are necessary; one truck dispatching schedule and one traffic light schedule. The quality of each of the schedules is

dependent on the content of the other, which makes the problem complex.

1.2.2 Real-time scheduling

In reality, many mines operate continuously day and night. In order to be able to compare the productivity and limit the simulation time, the time that is simulated as a work shift in this problem is set to approximately eight hours (500 minutes).

The simplest way of managing the schedules would be to generate them once in the beginning of a work shift and run the mine according to the set plan for the full eight hours. However, in reality there is a certain level of randomness incorporated into the problem. Some of the conducted operations differ in the time they require to be executed, one example of this being the time it takes to load and unload ore onto trucks. This randomness will cause discrepancies between the actual state of the mine (i.e. the current positions of the trucks) and what would be expected for each point in time (a beforehand predicted state of the mine). Allowing these discrepancies to accumulate would eventually lead to the schedule no longer being the best possible. Due to this, the schedule needs to be frequently updated - real-time scheduling is necessary.

The deviations from the expected operation times will not be large enough that the schedule would need updating every second or minute. Therefore, an updating frequency of 15 minutes is instead implemented. Every 15 minutes the software discards the old schedules and develops brand new schedules based on the current state of the mine.

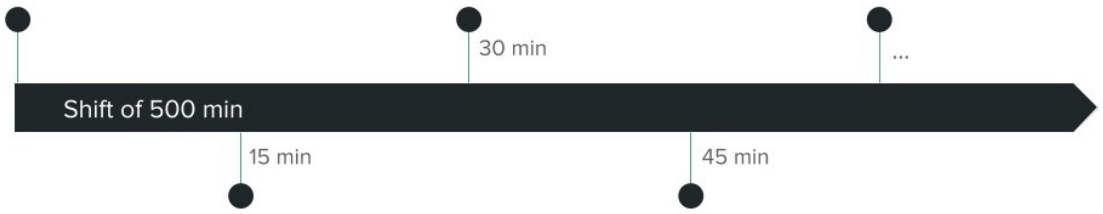


Figure 1.2: Updating frequency during work shift. Full shift of 500 minutes with schedule update occurring every 15 minutes.

1.3 Research question

To address the constraints and incentives associated with underground mining, and with the vision of delivering value to those operating in the industry, the objective of this study is to answer the following research question:

Can information from a previous run of an evolutionary algorithm be used when creating generation zero in the following run of the evolutionary algorithm in order to improve its performance?

The research question incorporates terminology that will be explained in Section 2.1. The objective of "improving performance" is vague and can be concretised by being broken down into the following three sub-meanings:

1. To increase overall productivity - An increased level of peak performance, i.e. maximised productivity in terms of number of truckloads being delivered to the crusher during a work shift.

2. To achieve better productivity when using a smaller number of trucks - Reaching a higher number of truckloads in cases when the number of trucks is lower than in a saturated state (minimum number of trucks in order to reach 99% of peak performance) of the mine.

3. Less computational power required to run the evolutionary algorithm - A smaller number of generations required to be run through by the evolutionary algorithm to find a solution, and thus less computational power and time required.

1.4 Limitations

Some simplifications that may affect the reliability of the outcome have been made.

In spite of the fact that the fitness function should be a measure of the metric that is being optimised, using the overall objective as the fitness function of the EA (maximising the number of truckloads) is in this case not feasible due numerous reasons. This was first discovered by Cox *et al.*, and several motives for working with a proxy metric instead were established [7]. For instance, the discreteness of the overall goal requires a full shift of 500 minutes to be simulated (at least), which in turn leads to drawbacks in terms of requiring longer run times as well as longer solutions (i.e. larger search space) compared to what could be achieved with a proxy metric. For the full discussion see [7, p. 12-13]. Therefore, a proxy metric is used as the EA's fitness function. The proxy metric is closely linked to the performance of the overall objective, but it is not exactly the same. Thus, it constitutes a limitation to the study. A more in-depth description of the proxy metrics can be found under Section 3.4.1.

Second, a number of factors observed in real-world mines have been omitted in the setup of the mine simulator. These are the same as mentioned by Cox *et al.*:

- Because of limited communication in the mines it is sometimes impossible to know the state of the mine
- Some systems have incorporated techniques for collision avoidance
- Some situations will require priority rules for different trucks
- Some mines operate over-trucked in order to always maximise productivity

Furthermore, the simulations in the experiments run for 500 minute work shifts. In real life mines the machinery usually runs continuously, only stopping for maintenance when necessary. It is however assumed that the mines will reach steady state during the time used in the simulations, thus, that the results are valid as a base of comparison.

1.5 Contributions

During the course of the project Hedda Malmström and Sofia Tenerz have worked closely together, both contributing to each section of the final product. Malmström and Tenerz were both physically present for the experimental phase of the project that was conducted in Perth, Australia during a period of eight weeks, as well as the following phase in Lund, Sweden, documenting the results.

This study, investigating the possibility to improve the performance of an EA, seeks to deliver value to the team of researchers at UWA, and thus in turn contribute to the industry of underground mining in Western Australia. Additionally, with the ambition of establishing conclusions relevant to EA applications in a broader context, this would ideally also mean contributing to the EA community as a whole.

1.6 Report outline

This report is structured as follows:

1. Introduction: Explains the general purpose of the project and gives an initial overview and understanding of the problem at hand. The research question is defined and limitations and contributions are investigated.

2. Background: The chapter is meant to explain all the background needed to comprehend the contribution made by the authors. First, general theory around artificial intelligence and EAs is explained. Second, a comprehensive literature review is conducted and lastly, the previous work made by the research group at UWA is examined.

3. Approach: The Approach chapter goes into detail on the existing problem and simulation setup as well as the implementation of the changes and how the experiments were conducted.

4. Evaluation: In Evaluation, the methodology is briefly explained followed by the results from the conducted experiments and related analysis. Graphs and tables provides an overview of the experiment outcome.

5. Discussion: In Discussion, the results are put in a greater perspective, analysing the contribution to academia and future research.

6. Conclusion: The results and discussion are compiled into a conclusion that provides an answer to the research question.

Chapter 2
Background

This chapter will first explain the theoretical concepts related to this area of research, which build the foundation in this research project. Thereafter, a literature study is compiled, exhibiting related work in fields from EAs to vehicle dispatching scheduling, as well as the two combined. Finally, the one study by UWA, which this research project builds directly upon, is described in closer detail. This to establish the starting point of the project and the surrounding prerequisites.

2.1 Theory

The theory behind this line of research is centered around the concepts of artificial intelligence, machine learning and more specifically EAs. This chapter explores the general definitions of these concepts.

2.1.1 Artificial intelligence and machine learning

Artificial intelligence can be defined as follows: *The effort to automate intellectual tasks normally performed by humans.* Artificial intelligence is a wide concept, and includes a plethora of various approaches that aim to make computers think the way humans do [6, p. 4-5].

Machine learning is a subset of artificial intelligence and is a collection of approaches and algorithms that are based on the computer independently learning and developing knowledge without explicit rules written by humans. In classical programming, humans would construct the rules that the computer uses to make necessary calculations in order to reach an answer. In machine learning, the computer is fed with data, and will construct rules or patterns on its own. Some machine learning systems use large data sets to be trained. As such, these systems builds mostly on empirical results rather than theoretical [6, p. 4-5].

2.1.2 Evolutionary algorithms

An evolutionary algorithm is one of many examples of a machine learning system. EAs aim to mimic the natural behaviour of evolution by iterating over several generations and using mutation and cross-over to produce offspring that combine qualities from their parents with new traits [5, p. 38-40].

In general, all EAs share three main characteristics: (1) They build on maintaining a population of different solutions to the same problem, (2) offspring are created using random operations made to mimic genetic mutation and recombination, and (3) every solution in the population will be evaluated using some kind of fitness function [5].

EAs are primarily a good choice when the search space considered is large enough that it would be unfeasible to try all solutions [28], and at the same time is uneven or unpredictable [5]. Figure 2.1 shows an example of a complex search space with several local optima. A simpler algorithm might get stuck and deliver a bad solution, but an EA has a good chance of performing well in spite of these circumstances [28]. As with all algorithms of this type one can never be sure that the optimal solution can ever be found, but the goal is to find a solution that is good enough for the particular situation in a reasonable time frame [28].

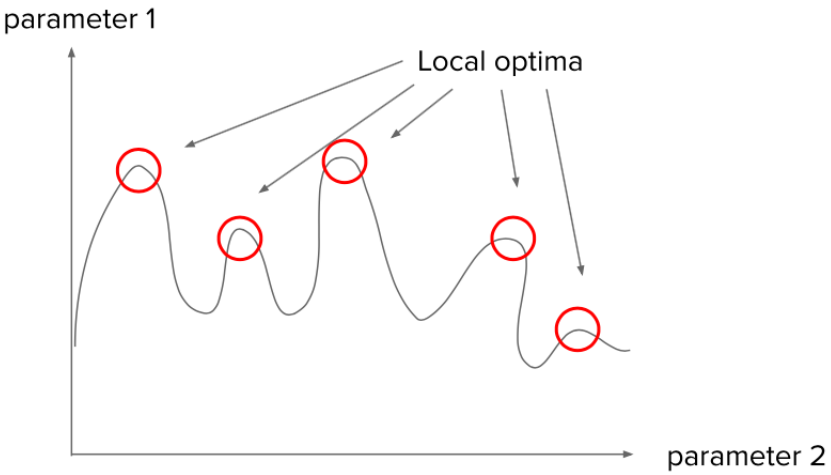


Figure 2.1: Visual representation of a complex search space.

The following two sections describe some of the terminology that is used throughout the report as well as the general concept behind the workings of an EA. Most implementations will be customised to the specific application in question, but the basic concept will in most cases be derived from this approach. The given information is based on the content of the course CITS4404 Artificial Intelligence and Adaptive Systems, UWA and is summarised in lecture slides [28].

Useful terminology

- The **search space** is essentially the landscape of possible solutions. Given a finite number of parameters one can sometimes visually represent the search space to get a better understanding of the size and complexity of it (see Figure 2.1).

- A **solution** is something that fulfills the requirements and objectives of the problem at hand; a particular way of getting from A to B. A solution needs to be represented in a way that can be understood and evaluated (by a fitness function, see below). In practice the **solution representation** could be a string of different decisions. As an example, imagine a network of roads between two points, start and finish. A 0 represents turning right at the next intersection, and a 1 means turning left. One way of travelling between the start and finish could look like this: [0 0 1 1 0 1 1 0].

  The solutions are sometimes referred to as 'parents' and 'offspring' depending on the current perspective of the process. Parents and offspring are both of the same solution representation, but the parent is the predecessor to the offspring.

- A **population** consists of a number of solutions. They are all solutions to the same problem. The size of the population will usually be kept at a constant number.

- **Generations** are, just as in biological evolution, the time variable of populations. In EAs the same solution will never change. After evaluation of parent solutions, some of the same solutions might survive and offspring will be created. Together they will make up the *next* generation.

- To create offspring some **genetic operators** are required. **Mutation** is essentially a slight alteration of the parent solution. This could for example consist of insertion or deletion of genes. Building on the example above, an insertion mutation could look like this (inserting a zero and removing the last gene, everything else remaining the same):

  $$[0\ 0\ 1\ 1\ 0\ 1\ 1\ 0] \Rightarrow [0\ 0\ 1\ 1\ 0\ \mathbf{0}\ 1\ 1]$$

  Crossover could for instance be choosing a sequence of genes (e.g genes on position 3-5) and switching places with the corresponding genes in another solution.

- The **fitness function** is used to evaluate the solutions. The fitness function is the same as, or closely related to, the objective of the problem. If the aim is to get from start to finish as fast as possible, an appropriate fitness function could be minimizing the travel time in seconds.

- After the fitness evaluation, the **selection** will occur. This determines which solutions that qualify for the next generation.

General concept

There is more than one way to construct an EA. In Figure 2.2, a simple illustration is made to explain the general concept.

1. An initial population (generation 0) is generated. The population size is constant (here: 4). The solutions in the first generation could either be constructed at random, or by some prior knowledge. The solutions have different shapes to symbolise that even though they are solutions to the same problem, the have different characteristics.

2. The fitness function is applied to all solutions and the fitnesses are compared to each other.

3. The best half (here: 2) gets to survive to the next generation and make offspring. This is an example of a maximizing problem. Hence, the two solutions with the highest fitness are the ones that are considered best.

4. Offspring are created with mutation and crossover from the chosen parents.

5. The former parents and the offspring they created make up the next generation (1...n). All solutions are equal and the process will start over from (1) again.

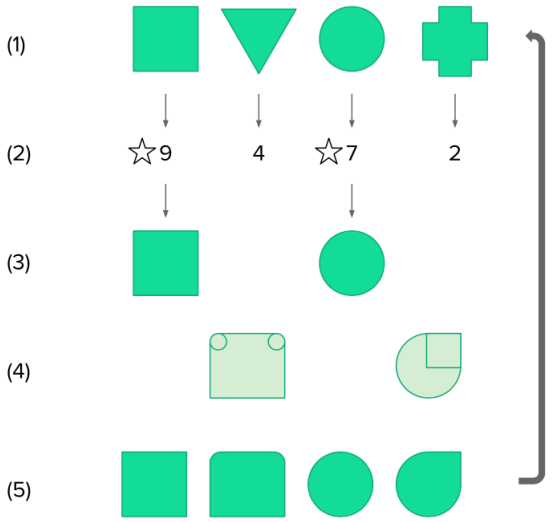


Figure 2.2: General Concept of an evolutionary algorithm. The different symbols do all represent solutions to the same problem, but with slightly different characteristics.

Termination of the process can be determined in different ways, the number of generations the algorithm will run for could be a constant number, or the algorithm can run until the improvement from one generation to the next has stagnated.

2.1.3 Co-evolutionary algorithms

Co-evolutionary algorithms, henceforth referred to as CEAs, are an extension of the simpler evolutionary algorithms. The idea builds on combining different types of solutions from separate populations that together make up the complete solution to the problem at hand. This will make it possible to apply EAs to more complex problems and explore larger search spaces [21].

In Figure 3.2.2 the concept is illustrated.

1. Two generations are maintained, both containing four different solutions at each point in time. They represent different parts of a complete solution and both are needed in order to form the full solution to the problem at hand.

2. Members from the two populations are paired to evaluate the fitness. They are both given the same fitness score based on the quality of the solution they together made up.

3. After the fitness evaluation the algorithm will perform selection of the best solutions. After this, offspring are created the same way as in an EA; using genetic operations on the surviving parents.

4. This process continues for several generations until termination.

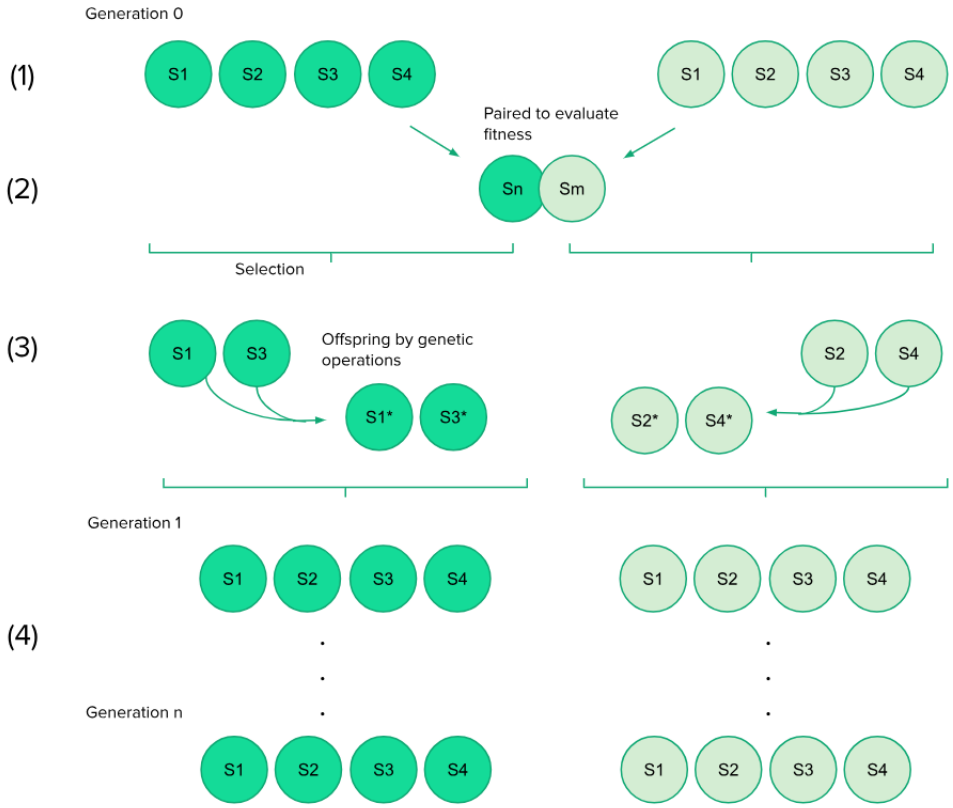


Figure 2.3: General Concept of a co-evolutionary algorithm.

2.2 Literature review

There is much work related to this research in one way or another. Very closely related work, using the same type of algorithm for the same problem, is however limited. Nevertheless, using EAs for adjacent problem setups is frequently occurring. In addition, another commonly researched area is the very same problem, scheduling in mines, being solved by other means than this specific type of algorithm. The purpose of this section is to summarise relevant related work and highlight and discuss potential takeaways that could be valuable going forward.

2.2.1 Evolutionary algorithms for truck dispatching in mines

The use of EAs to target the issue of scheduling in underground mines is a niched matter, which seems to have been exposed to limited amount of research. The group dedicated to this at UWA have established findings on multiple occasions. Cox *et al.* proved 2017 that using EAs (there referred to as genetic algorithms) for this purpose displays benefits compared to both the industry standard DISPATCH as well as the simpler approach of using greedy heuristics [9] [24]. Benefits were seen in terms of production increase and cost-savings, and mainly made a difference in setups where other approaches struggled, for instance when one-lane roads were included.

The research was thereafter taken further by the development of a cooperative CEA that performed real-time evolution of both dispatching schedules and traffic light schedules together [8]. The problem setup here also included one-lane roads, which had previously been seen as the main area for potential improvement. Final results established that the CEA outperformed other approaches, primarily by delivering high productivity with fewer trucks in use. These findings were covered on a brief level on the Australaisan Joint Conference on AI in 2018. More exhaustive documentation of the proposed approach as well as adjacent experiments was compiled into an article submitted (but not yet published) in late 2018 [7]. This is the article that has been considered the starting point for this study, being described in closer detail in Previous work (Section 2.3).

The only other identified case that investigates the use of EAs for truck dispatching in mines was performed by Mendes *et al.* in 2016 [16]. Documentation was done solely in Portuguese. Knowledge and takeaways gathered from this should therefore be treated with care as there is a risk of translation fallacies. In this study, Mendes *et al.* implement and compare different versions of a hybrid multi-objective EA (hereafter referred to as hMOEA). This study is very similar to the ones that have been conducted at UWA, primarily considering the fact that they both use real-time dispatch scheduling and evaluate the solutions by simulating in abstract mine setups where trucks travel between shovels and crushers.

In the study conducted by Mendes *et al.*, each approach is evaluated by calculating Inverted Generational Distance (IGD) and hypervolume. This also makes the objectives similar to previously described work, where the main performance indicator also relates to productivity in terms of volume of ore delivered. The research performed by Mendes *et al.* primarily builds upon the following two concepts, which constitute the differences between the hMOEA versions compared:

1. Implementing a local search engine, a heuristic, for constructing the first generation

2. Applying a Pareto local search method for multi-objective problems (MOPs) to the set of generated feasible solutions to enable faster identification of a new non-dominated solution [30]

In this study, the multi-objective nature of the problem is emphasised and several more variables are introduced (compared to the problem setup in the work done at UWA). This further complicates the problem and raises the level of difficulty in finding a suitable solution. For example, the trucks used in the mine all differ in loading capacity.

Introducing local search methods and thus achieving faster convergence is desirable in complex problem setups [1]. However, considering the relevance of this work in relation to ours, one can conclude that these findings might be less applicable to our project due to the fact that our problem setup is more simplified. For instance, in the case of generating dispatching schedules (our starting point) the solution representation is basic, meaning adding local search engines will deliver limited value. For the development of traffic lights schedules on the other hand, where solutions have more room for variation, this concept might be of interest at a later stage.

2.2.2 Evolutionary algorithms for other forms of mine planning

Even though work done investigating the use of EAs for truck dispatching is limited, the use of EAs in adjacent fields seem to have been more commonly addressed. Bitanshu demonstrated in 2012 the advantages of using EAs for open pit mine production scheduling, i.e. planning the extraction sequence of mining blocks [11]. Solving the problem of production scheduling in an open pit mine means solving a large scale mixed integer programming problem with a large search space, whilst being subject to constraint equations. This could, according to Bitanshu's research, be done in significantly shorter time while still finding satisfactory solutions when using an EA.

Optimising production scheduling using EAs seems to be an anticipated future lucrative business, with mine consultancies developing commercial algorithms. An example of this is the EvORElution, a trademark software optimising open pit mine scheduling using an EA [18]. This has been developed by the Western Australia-based mine consultancy ORElogy. Escalating to a higher level of planning, EAs have also been introduced to multi-mine planning, i.e. planning across several mine sites [19]. This has shown potential benefits, but since the study investigating this was the first of its kind (or one of very few) the main conclusion was that further fine-tuning and extention of scope was to be done.

2.2.3 Evolutionary algorithms for vehicle dispatching problems

The problem of truck dispatching scheduling in mines show both similarities and differences to vehicle dispatching problems in general. Alarie and Gamache describe two typical aspects of a vehicle dispatching problem:

1. The variability of the demand
2. The multiple objectives [2]

Difficulties and complexities related to the problem are subsequently based on these aspects. There are also challenges indirectly related, as for instance determining the size of the fleet (i.e. depending on demand). Typical for these problems is that they are often too complex and include too many parameters and constraints to be solved by deterministic methods. Therefore, heuristic solutions have traditionally been seen as common practicable methods for solving the problems. Heuristic approaches are described in closer detail in Section 2.2.4.

The truck dispatching problem in mining can be seen as a simplification of the general vehicle dispatching problem. It treats a closed system (pick-up and drop-off points stay the same over time), trucks are loaded with ore to full capacity at one shovel and then sent directly to a crusher (meaning there are never several pick-up points on any one route), time of travel is short compared to the full shift and visiting frequency to every pick-up point is high; to name a few significant differences. Furthermore, the near future of the state of the mine can be predicted and the level of randomness incorporated to the problem setup is limited. This opposed to the general case, which usually has a stochastic nature at any moment. These differences in overall characteristics have to be kept in mind when taking part of methodology or algorithms developed to address truck dispatching problems in general or for other industries, for the purpose of our research.

Potvin has compiled a review of the work done on EAs for vehicle routing problems (VRPs) [22]. For the study at hand the subcategory of vehicle routing problems considering time (time-dependant VRPs), rather than distance travelled, is the most relevant. This because time-dependant VRPs display the greatest correlation to both the overall goal and the proxy metrics of our study (described under Approach, Section 3). In the work studied by the author some shortcomings to the classical EA were encountered when applying it to VRPs. Many of these shortcomings matter exclusively when considering the conservative and classical view of the EA. When moving away from this static view of the features of the algorithm (as in the case of the EA developed by UWA and used in this study) many of these shortcomings are already acknowledged and regarded for. However, one mentioned shortcoming that still stands is the notion that heuristic information has to be incorporated into the algorithm to ensure the EA obtains competitive results. Since this also aligns with what was stated by Mendes *et al.* [16], but does not seem to be an established truth in the work by UWA (which is the starting point for this study) [7], this should be kept in mind going forward.

On an overall level the review conducted by Potvin displays an extensive use of EAs to address vehicle routing problems [22]. Looking back, research on this has chiefly been conducted for the generic case, but is seen increasingly for niche problem configurations as well as real-world cases. Due to the flexibility of EAs in terms of problem setup constraints, real-world application will most likely continue to increase ahead. Dynamic problems requiring fast response times (as for instance in the case of real-time scheduling) is another area where the use of EAs is expected to grow. This is due to the fact that former limitations related to the high computational burden of EAs are being eradicated with technical development.

2.2.4 Other methods for truck dispatching

There is much research being done on truck dispatching in mining. This can be interpreted as an indication of growing enthusiasm for more advanced methods for truck dispatching in commercial circles and real-life mines as well. Many authors agree that transportation costs make up as much as 50-60 percent of total operations costs in mines [2], which is why a growing interest in performing these activities more efficiently is highly motivated. Work regarding truck dispatching has been geared towards both open-pit and underground mining, with the former constituting a clear majority of the performed research. Open-pit mines have used dispatching systems for the last 45 years [14]. The problem of truck dispatching in an open-pit mine is simpler since it does not have to consider routing and scheduling aspects in

the same ways as in the case of underground mines. Even so, methodology developed around open-pit mining could provide valuable insights also to the context of underground mining.

Munirathinam and Yingling have published a comprehensive review of the currently existing computer-based truck dispatching technologies for open-pit mines [17]. These are classified into the following overarching categories:

1. Systems that employ heuristic rules to make truck assignment decisions
2. Plan-driven dispatching systems
3. Constrained-assignment dispatching systems

Conclusions drawn from the survey establish that while systems based on heuristics are easy to implement, they are limited in their contribution. Due to their simple nature, they do not fully capture the multi-objective essence of the problem. Despite this, Munirathinam and Yingling also claim that when applied in the right contexts, heuristic rules can be a superior dispatching methodology. For instance, one key area is that of very complex problems, where other methods need to simplify the problem a lot to make it graspable, compromising appropriateness and/or effectiveness. Another area where heuristic rules are effective is in contexts that are very prone to unpredictable fluctuations, which can be the case of large and complex mining operations.

The other two approaches to truck dispatching, plan-driven and constrained-assignment dispatching systems, are both considered highly beneficial, and display many different advantages respectively. However, the authors have identified a lack of studies comparing the plan-based and constrained-assignment approaches, so that declaring a winner among the two is difficult.

Alarie and Gamache also provide a review of truck dispatching strategies in open-pit mines [2]. They conclude their paper by listing the characteristics that are considered to make up the ideal truck dispatching systems. Also here the plan-driven approach is brought forward, in this article referred to as the "multi-stage approach". In short, the very core of this approach is the two-plan structure [17]. The first component deals with short-term production planning, while the second component deals with the dispatching procedure used to make truck assignment decisions in real-time simultaneously.

Both White and Olson and Temeng *et al.* have developed mathematical programs that build on the plan-driven approach [29], [27]. The mathematical two-stage dispatching program developed by White and Olson is packaged as the trademark software "DISPATCH". DISPATCH is today commercially available as DISPATCH Fleet Management System, with the most recent version said to mathematically build upon the three pillars of linear programming (LP), dynamic programming (DP) and Best Path (BP) [25]. DISPATCH has also been updated to include real-time scheduling. Moreover, an adaption of DISPATCH now exists that is specifically developed for the use in underground mines, offered by the same supplier [24].

Truck dispatching strategies developed for *underground* mining specifically is, as mentioned above, a less exploited field of research. Gamache *et al.* present a graph-based approach to this built upon the Dijkstra's shortest-path algorithm [14], [12]. Ozkarahan *et al.* on the other hand have developed a mathematical model based on Mixed Integer Programming (MIP) to solve this scheduling problem [20]. The problem is here modeled as a parallel machine scheduling problem where MIP is employed to find a solution.

Looking into a closely related field of research, scheduling automated guided vehicles (AGVs) in manufacturing shows many similarities to scheduling trucks in underground mines. Kim and Tanchoco present a method for conflict-free shortest-time AGV routing, a method that seems well suited also for underground mine scheduling [15]. This is primarily due to the use of bi-directional road segments, which makes the context similar to that of underground mines with one-lane road sections. The algorithm proposed by Kim and Tanchoco is also based on Dijkstra's shortest-path algorithm [12].

As mentioned above, the most simple computer-based truck dispatching strategies build on heuristic rules [17]. For a system like that, one truck is dispatched at a time, considering nothing but the one greedy rule put in place. Examples of greedy heuristic rules for this purpose, that have been examined as potential rules for deciding truck assignments in mines, are the following:

1. Fixed truck assignment - each truck is assigned to one shovel only, and stays on the same loop between the crusher and the one assigned shovel.

2. Minimising truck waiting time - a truck is dispatched to the shovel where it has to wait the least time. This is achieved by minimising the difference between the time for the shovel to get ready (finish loading of all trucks queued up) and the travel time for the dispatched truck.

3. Maximise truck - a truck is dispatched to the shovel where it can expect to be loaded with ore at the earliest future point in time. This builds upon the greedy rule above (minmising truck waiting time) with the addition of the time for the shovel to load the truck.

4. Minimising shovel waiting time - A truck is dispatched to the shovel that has been waiting the longest, or that is expected to be available the soonest. This strategy is also referred to as maximise shovel rule.

5. Maximising truck momentary productivity - A truck is dispatched based on where it will maximise its truck momentary productivity. The truck momentary productivity is defined as the ratio between truck capacity and truck cycle time.

6. Minimising shovel saturation - A truck is dispatched to the shovel with the least saturation. Saturation is defined as the ratio between the number of trucks assigned to that shovel and the number of trucks that should ideally have been assigned to that shovel. This ideal number, referred to as the saturation number, is the number of trucks given by the ratio of the travel time between the dispatching point and the shovel and the servicing time per truck at the shovel.

7. Minimising deviation from shovel production target - A truck is dispatched to the shovel that is lagging behind its output targets the most.

The seven heuristic rules briefly described above are the most common ones, which have been investigated and evaluated in many different studies. For more ingoing descriptions and analysis, thorough work has been done by both Munirathinam and Yingling as well as Tan and Ramini [17], [26].

2.3 Previous work at UWA

This research project builds directly upon previous research conducted at UWA, and specifically on the most recent work by Cox *et al.* [7]. In this, the authors seek to examine the use of EAs and CEAs for real-time dispatching and traffic light scheduling in underground mines. This section describes the design of the experiment and the EA as it was developed by Cox *et al.*, as well as the key findings. This constitutes the starting point for our study. Further development of the EA as well as experiment setup specific to our project is elaborated on under Approach, Section 3.

2.3.1 Experiment design

The study evaluates the use of evolutionary (and co-evolutionary) algorithms in three different contexts, with the overall mine set-up being the same as described in Section 1.2:

- Using an EA to develop the dispatching schedule, with traffic lights regulated by simple rules or fixed schedules.
- Using an EA to develop the traffic light schedule, with dispatching decisions regulated by simple rules or fixed schedules
- Using a CEA to evolve both the dispatching schedule and traffic light schedule together

When traffic lights were regulated by simple rules, greedy rules were applied (also called greedy heuristics or heuristic rules). Traffic lights regulated by fixed schedules meant regulation by cyclic timers. This was in the documentation referred to as greedy lights (GL) and cyclic lights (CL) respectively. The performance was assessed in terms of mine productivity, i.e. the number of truckloads delivered to the crusher during a shift of 500 minutes. Evaluating fitness of solutions produced by EAs was done using proxy metrics. The three proxy metrics used were:

- Minimise total truck waiting time (W)
- Minimise average truck cycle time (C)
- Minimise average crusher inactivity (I)

These proxy metrics are explained in closer detail under Section 3.4.1.

The three developed approaches were comprehensively compared to other truck dispatching technologies. Inspiration for these dispatching technologies was drawn from related academic work as well as solutions currently in use commercially. The three benchmark scheduling tools that were chosen were:

- Best-H - a portfolio of seven commonly used greedy heuristics
- DISPATCH - an industry standard approach based on linear programming
- FCS - a flow-based approach based on linear programming

2.3.2 Findings

Experiments were executed stepwise. First, it was observed that from the set of existing benchmark approaches Best-H performed the best, which it did in combination with controlling traffic lights by cyclic timers (CL) (see Figure 2.4). Performance is here measured by truckloads of ore being outputted over a shift, i.e. productivity of the mine.

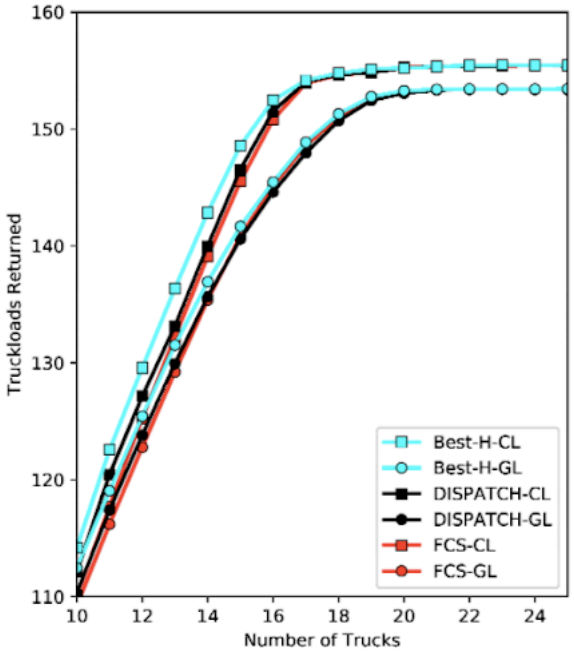


Figure 2.4: Comparisons between several benchmark approaches to dispatching scheduling, with traffic lights regulated by cyclic timers (CL) and greedy rules (GL) respectively. Averages of six problem instances. Graph indicates that Best-H with cyclic timers generates the highest productivity, i.e. number of truckloads.

When this was determined, Best-H was subsequently compared to all of the three EA and CEA approaches developed by Cox *et al.*, combined with both CL and GL traffic light regulation. What could be observed was that using EAs to generate the dispatching schedule outperformed Best-H when using GL to regulate the traffic lights, but did not yield any remarkable results when being combined with CL. Furthermore, using the EA to develop traffic light schedules while keeping the dispatching schedule fixed, did on a general level not perform better than Best-H.

However, the most prominent finding from this research were the results produced by the CEA, seen in Figure 2.5. The CEA outperformed all other approaches, no matter what proxy metric was used for evaluation. The biggest difference was seen in achieving a high mine productivity with a low number of trucks, as well as reaching maximum productivity with fewer trucks.

This study produced very satisfactory results, and left lots of openings for further investigation in future research. The article documenting this research concludes by suggesting future paths of research within (1) modification and refinement of the problem setup, and

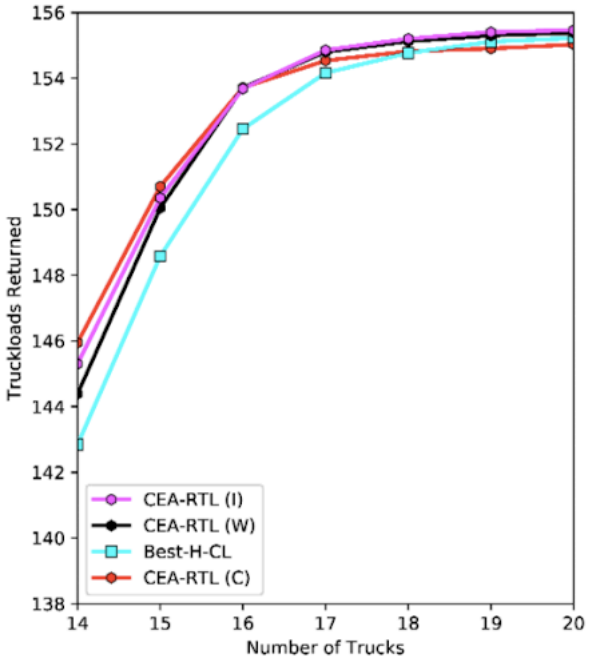


Figure 2.5: Comparisons between the BEST-H-CL and CEA evaluated using three different proxy metrics. Averages of six problem instances. Graph indicates that the CEA performs better than Best-H for all three proxy metrics in use.

(2) modification to the EA. Our project is based on the second of these two suggestions - modifying the EA to further enhance its performance.

Chapter 3
Approach

This chapter elaborates on the problem description from Section 1 by providing a closer look into the scheduling problem and its intrinsic features and limitations, as well as the details of the EA and CEA solutions. Thereafter, the suggested changes to the EA and CEA that constitute this study are described, first as an overall theoretical concept and then as the multitude of practical variations that were implemented. Lastly, some of the decisions made around the experiment setup are disclosed. All information building up the section below is obtained from three studies performed at UWA unless otherwise stated [7], [8], [9].

3.1 The scheduling problem

In the following section the simulation model is explained. Thereafter, the nature of the scheduling problem is described, which is highly dependent of the setup of the simulator. Finally, the concept of schedule management is covered, exhibiting the reason for real-time scheduling and what that means in this context.

3.1.1 The simulation model

In order to compare the different approaches for scheduling in underground mining, a simulation model was used. This model is the very same as was originally constructed for work done by Cox *et al.* [9], and subsequently used for further experiments by the same team [8] [7]. The simulator is designed to represent an underground mine site, and is constructed based on a network of timed automata (TA) [3]. Further documentation of the theoretical concepts forming the foundation of the simulator can be found in the original article [9].

The simulator, designed to correspond to a real-life underground mine, has the overall structure as depicted in Figure 3.1. It consists of four shovels on different depths. A truck gets a destination shovel at the time of departure from the crusher. At the shovel, ore is loaded onto the truck. Thereafter, the truck travels the same route back from the shovel to

the crusher at the surface, where the ore is unloaded. The downward sloping road sections are one-lane, and the road sections branching off towards the shovels are two-lane. Which truck gets to traverse each one-lane section at each point in time is regulated by traffic lights. For each one-lane road section, there is a pair of traffic lights that need to be regulated by a schedule. When one of the two is green, the other is automatically red to ensure that collisions will never occur.

These two components make up the essence of the scheduling problem; the scheduling of the trucks' destinations and the timing of the traffic lights, designed to achieve the highest mine productivity possible. The overall goal, maximising the mine productivity, is measured in number of truckloads being delivered to the crusher during a shift. A shift is set to be 500 minutes, i.e. just over eight hours.

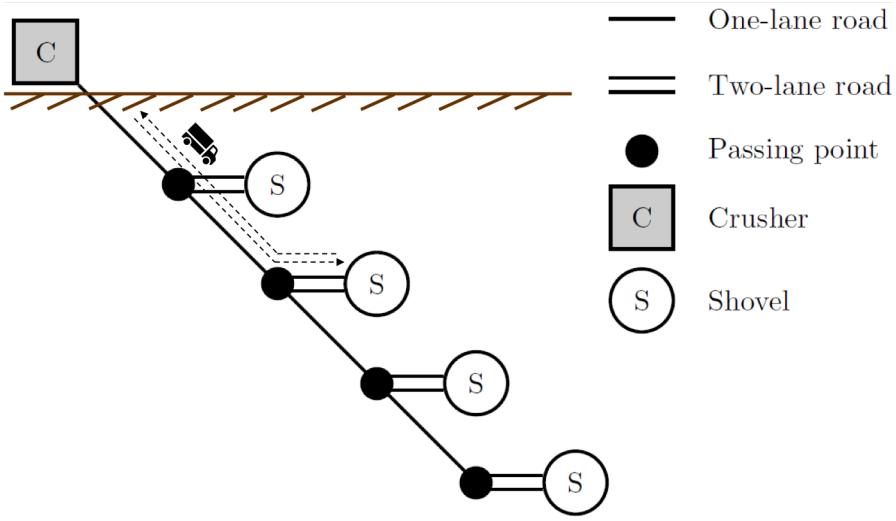


Figure 3.1: Model of an underground mine. A truck is given a destination shovel each time it leaves the crusher. Access to each one-lane road is controlled by a pair of traffic lights.

In this mine setup trucks are homogeneous, meaning they all have the same loading capacity and speed distribution. The trucks vary in their average speed depending on their state (empty or full) and trajectory as in Table 3.1. The crusher and shovels are heterogeneous, they vary in average service rates among themselves and depending on the problem instance, see Problem instances below.

Table 3.1: Trucks' average speeds.

trajectory	contents	speed (km/h)
level ground	empty	15
level ground	full	12
downhill (on decline)	empty	15
uphill (on decline)	full	6

Problem instances

Six different problem instances were generated for the original mine simulator, all with four shovels and one crusher. For each problem instance, the lengths of the road sections as well as the average service rates were generated randomly from the following ranges:

- Downward sloping road sections: 400 - 600 m
- Crosscut road section: 150 - 250 m
- Average filling rate at shovel: 9-17 minutes per truck

The average filling rates for all shovels together make up to the emptying rate at the crusher, which is three minutes per truck. Thereby, the shovels and the crusher could all theoretically work continuously and never stand idle. Shovels and crusher will never be held back by one another, the bottleneck will always be the trucks and traffic lights. Inspiration for the mine setup was taken from a review by Rupprecht [23] in order to ensure staying well aligned with reality.

3.1.2 Schedule management

The unpredictable mine

There is complexity to this scheduling problem in regards to a certain level of randomness being incorporated in the simulation model. Activities performed in the mine differ in terms of time required for execution, to better reflect corresponding real-world activities. Times for loading and unloading the trucks with ore, travelling the road sections and queuing for any of the activities, are all randomly generated. As seen above, times to perform these activities are specified with an average, but no further. Due to this, a future state of the mine is never fully predictable. The current state of the mine on the other hand, i.e. where all the trucks are and how far through the ongoing activity they have gotten, is always measurable. For any point in time, every single truck can be assigned an effective position in the mine by a state-value pair.

Real-time scheduling

When a schedule is developed, it is done based on a predicted future state of the mine. As time passes, a discrepancy will arise between the predicted state of the mine and the state of the mine as it actually is. The larger this discrepancy, the worse suited the schedule is for the real state of the mine. Therefore, the schedule needs to be updated to find one that is optimised for the current state of the mine, i.e. find a new solution customised to the problem setup as it looks at this specific point in time. When it is time to update the schedule (either truck dispatching schedule, traffic light schedule, or both), this is when the chosen scheduling approach comes into play.

As briefly explained in Section 1, real-time scheduling in this context refers to the practice of updating the schedule/-s regularly. Here, the update occurs approximately every 15 minutes. The update frequency is set arbitrarily and is potentially subject to further research.

3.2 Evolutionary algorithms to develop schedules

This section explains the structure of the implementation of the software used to apply EAs or CEAs to solve the problem explained above. As explained in Section 2.3.1, two different approaches have been used to tackle the scheduling problem. First, using an EA to schedule truck dispatching and using cyclic timers for lights, and second, using a CEA to evolve the two schedules together. The implementation was done as a part of the work done by UWA, and the same setup is what is being used as a base for comparison in this study.

3.2.1 Evolutionary algorithm

For the EA, a population of solutions to the truck dispatching problem is maintained. The population size is 100, which will be kept the same at all times. One wants the population size to be large enough that the search space is explored, but at the same time not too large which would cause the algorithm to be slow, hence 100 is a good compromise. The solution representation is a list of different shovel IDs (numbers ranging from 0 to 3) representing the order in which trucks will be dispatched to the different shovels. Each truck, no matter how many there are in use, polls its next destination from the same list when it has finished emptying a load of ore at the crusher and is about to begin a new cycle. Hence, a typical solution will look something like this:

[1, 3, 0, 2, 3, 0, 1, 2...]

For the first generation of the EA run, generation zero, the solutions will all be generated randomly.

The solutions are then evaluated by simulating over time H (see Section 3.2.3 below), and using the fitness function. The fitness function in use is *minimising the crusher inactivity time* and is further elaborated under Section 3.4.1. As such, the fitness returned by the fitness function is the portion of time the crusher is expected to stand still during the time H with the current solution in use, thus, the lower it is the better the solution. Because of the randomness incorporated in the simulator as mentioned above, a solution could potentially get "lucky" or "unlucky" in an evaluation. To ensure that each solution gets a fair fitness score, 20 simulations are completed for each one. The resulting fitness will be the average of the 20 simulation scores kept in a "fitness bucket". Upon creation, the solution is given 20 fitness evaluations and for each generation it survives (see below) one new evaluation is added to the bucket and the oldest one will be excluded.

An illustration of the process of creating the next generation is shown in Figure 3.2. After being evaluated by the fitness function, the solutions are sorted by quality. The simulator allows us to choose the elitism, the portion of each generation that is guaranteed to survive to the next generation, in this case only the best one. The entire population is then used as the base for reproduction and 100 offspring are created and put in a selection pool. The solutions from last generation that were not guaranteed to survive (all but the best one) will be put in the same selection pool, making the size 199. The selection is then made and the 99 (population size: 100 - guaranteed survivors: 1) best solutions from the selection pool are added to the next generation. Most likely, the new generation will be a mix of the best solutions from last generation and the offspring that were created with them as parents.

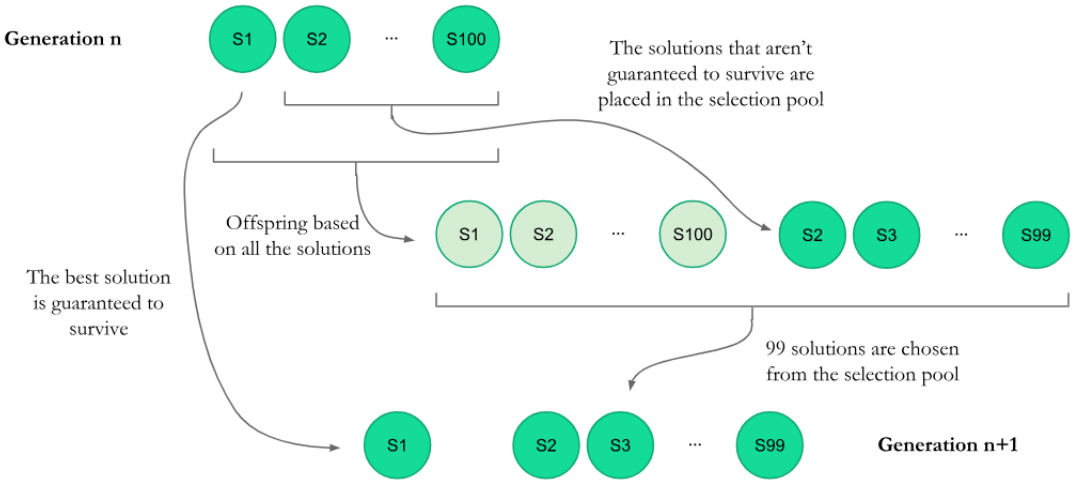


Figure 3.2: Evolutionary algorithm process.

The process explained above will continue until termination is caused by one out of two possible reasons. The first reason is improvement stagnation. When the improvement in fitness has been less than 0.5 percent for the last 100 generations there is presumably little value to be added by continuing the EA run, and thus the process is terminated. In cases when a large number of trucks are in use (see Section 3.4.3), a second reason for break is necessary; when fitness reaches the optimum value. Since the fitness function in this particular case is minimising and the crusher inactive time cannot be less than zero, there is no point in continuing to come up with new solutions as it cannot get better. The run is then terminated. The optimum will never be reached for a full shift, however, for the simulation window H (see Section 3.2.3), it can be.

The output from the EA run will be the solution with the best fitness in the last generation, and that will be put in use.

3.2.2 Co-evolutionary algorithm

The way the CEA works is in many ways similar to the EA. The big difference is however that two separate populations are maintained, one for truck dispatching schedules and one for traffic light schedules. The population size for both populations is 100. The solution representation for the truck dispatching schedule is the same as in the EA; a list of shovel IDs. The solution representation for the traffic light schedule is slightly different as it contains four different sub-schedules, one for each pair of traffic lights at each one-lane road section in the mine. Each traffic light sub-schedule is made up of a list of times (in seconds) that the specific light pair is supposed to alter between green and red. In the solution the four lists are combined into one long list, but when the schedules are to be used this has to be split into the four parts. Just as in the EA, the solutions in the initial populations are generated randomly.

To form a complete solution one member from each of the two populations is required, which is how the fitness is evaluated. Each member of the first population is paired with a number of collaborators (in this case one member) of the other population. The fitness that

the pair generates gets assigned to them both. The schedule-pairs are evaluated 20 times in order to reduce noise, and the resulting fitness is an average.

An illustration of the process is shown in Figure 3.3 below. For each of the two populations a selection pool is created and offspring based on the entire population is placed there. In this case, all the solutions from the previous population survive and are also part of the selection pool. After this, the fitness is evaluated as explained above, resulting in the two populations again being filled with the 100 best solutions in a following generation. The populations evolve separately from each other, only the fitness evaluation uses the combination.

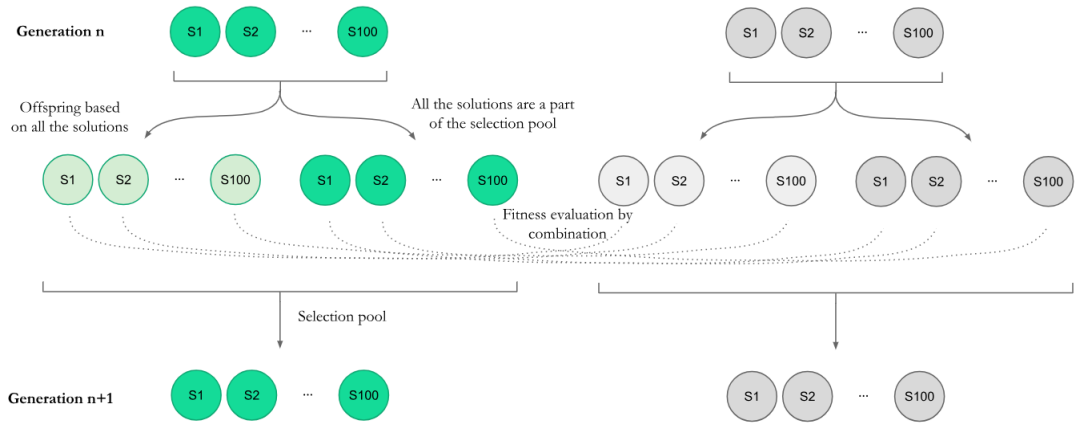


Figure 3.3: Co-evolutionary algorithm process.

The process termination works the same way as with the EA, either stopping because of improvement stagnation, or because the function has reached the optimal fitness. The output of the CEA run is the best observed pairing of schedules and they will together be put into use.

3.2.3 Simulation time

Considering the fact that each schedule will only be used approximately 15 minutes before being replaced (see Schedule Management, Section 3.1.1), there is no reason to ever develop schedules that cover a full shift. On the other hand, there are several reasons to develop schedules longer than exactly 15 minutes: first, software optimising a schedule that is only to be in use for *the next 15 minutes* sees no reason to leave the mine in a state that will provide a beneficial starting position for the subsequent 15 minutes. Nobody, not humans nor software, has the knowledge of what a "good" starting position would be, why the simplest way around this problem is to have the algorithm develop schedules that do consider the near future as well. Second, the simulator is built in a way that it does not always update the schedule/-s after exactly 15 minutes. For instance, the dispatching schedule will only have a chance to update when a truck is about to depart from the crusher, thus requesting a new shovel destination. This will occur after 15 minutes plus the additional time until a truck requests a shovel destination again. There are also other reasons that might delay the schedule update. To ensure developing schedules long enough to hedge for all of these potential delays, the schedules are developed to be the same length as the longest possible cycle time for a truck

in the mine, i.e. going from the crusher to a shovel and then back to the crusher again. This schedule length is called H and is approximately 60 minutes (varying slightly depending on the problem instance).

The overall goal, number of truckloads delivered to crusher, is difficult to measure when evaluating shorter time periods than the full shift (500 minutes). This due to the fact that the outputs are integers that differ very little between approaches (usually one or two truckloads). Thence, approaches evaluated over shorter time periods might output the same productivity while actually differing in performance. Using proxy metrics (see below, Section 3.4.1) when evaluating the schedules (solutions) is therefore another factor that enables developing shorter schedules of time H.

3.3 Using previous schedules

The mine simulator and the EA as it was developed up until the last research project at UWA, constituted the starting point for this project. It was from here that changes and add-ons to the EA were designed and implemented.

As stated before, a new EA or CEA runs every 15 minutes to make sure the schedule in use is up to date and not too affected by the randomness in the simulator. With the original setup this means that every 15 minutes the old schedule is discarded and new schedules are produced, starting off by randomly constructing the first generation of the next EA/CEA run.

However, considering the last generation of the previous run, it will contain the one solution that was taken into use, and 99 other schedules presumably nearly as good as the one that was chosen. This constitutes the main idea of this study; using the solutions from the last generation of the previous run to construct generation zero in the next run, and by doing so potentially improving performance of the algorithm.

When constructing the new schedules in the original approach, one of the key elements is taking the current state of the mine into consideration (where the trucks are positioned and how far along they are in their current activities) at the time of the update. If the solutions that are saved from the previous run do not take this into account the results would presumably be bad, as they are optimised for a different state compared to the one the mine is currently in. The key to this is to only use the solutions that would have resulted in the same state of the mine after 15 minutes, meaning the solutions that match the one that was actually chosen on the assignments that were completed before schedule update.

As explained in Section 3.2.3, the time that the solution is developed for is H (approximately one hour). The first 15 minutes that have passed by the time of the next update therefore needs to be cut off. In order to again develop schedules of the correct length (H) this will also mean that a new ending of the schedule needs to be added. The procedure is visualized in Figure 3.4.

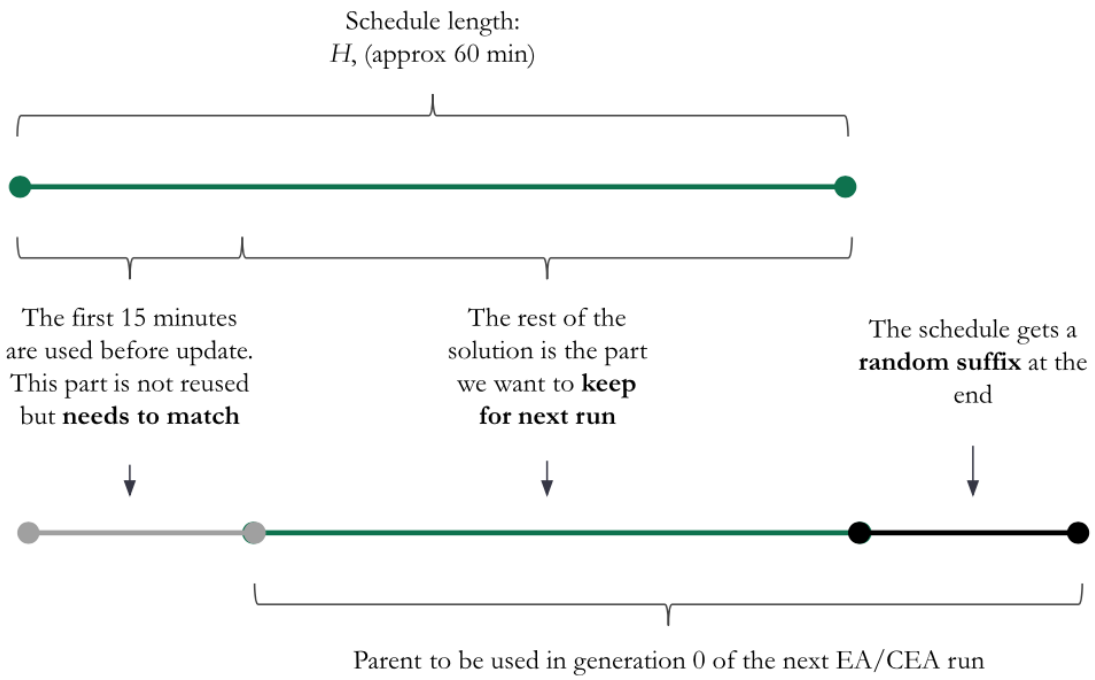


Figure 3.4: Matching and fitting schedules.

3.4 Experiment setup

Decisions regarding experiment setup, non-specific to the spectra of approaches developed, had to be made. Such are disclosed in the section below.

3.4.1 The proxy metric

As briefly covered in Sections 1.4 and 3.2.3, there are complications related to the EA optimising by the overall goal, i.e. maximising productivity measured in truckloads of ore being delivered to the shovel during a shift. Therefore, the predecessor to this study [7] examined the use of proxy metrics. A proxy metric is a metric that can be measured instead of the overall goal, and by doing so also indirectly measuring the overall goal. Subsequently, maximising or minimizing (depending on character) an adequate proxy metric will lead to maximising truckloads of ore being delivered to the crusher (as is the main objective). Three different proxy metrics were suggested and examined in the study by Cox *et al.* [7]:

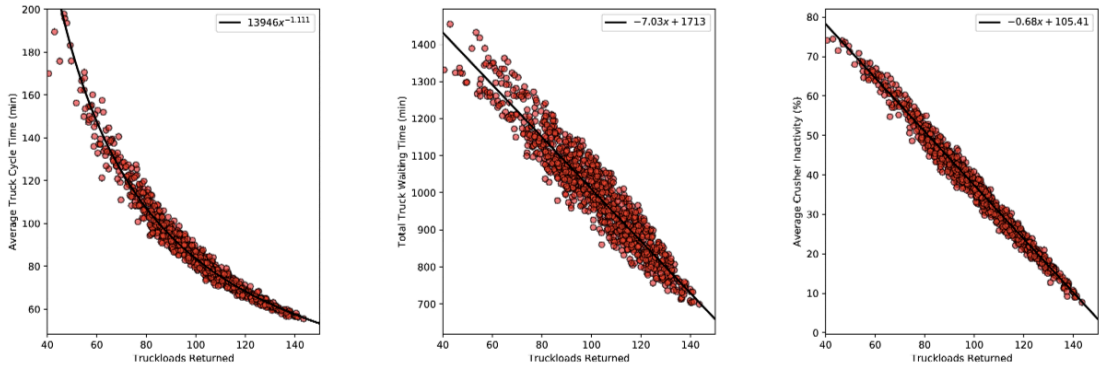
- Minimise average truck cycle time - Returns the average time for a truck to perform a full cycle, i.e. the time between getting an assignment at the crusher, performing the action and returning to the crusher. The simulation window is in this instance extended so that a truck that has left the crusher within the original simulation window H has enough time to return and thereby finalise its cycle.

- Minimise total truck waiting time - Sums up and returns the total waiting time for all trucks, i.e. time spent queuing at shovels, crushers and passing points. Has the

advantage that some effects of decisions can be observed immediately, thus a shorter time horizon can be used.

- Minimise average crusher inactivity - Returns the average fraction of time the crusher stands idle, i.e. is empty of ore to crush. This number is computed as the crusher idle time divided by the time of the simulation. Here as well, the simulations window is extended to wait for all trucks dispatched during the original simulation window H to finalise their cycle.

In order to validate the use of the three proxy metrics, the correlation was examined between the overall objective and each of the three proxy metrics respectively by Cox *et al.* [7]. The findings were compiled into the three charts in Figure 3.5.



(a) MATCT. (b) MTTWT. (c) MACI.

Figure 3.5: Plots demonstrating the correlation between total production during a shift and the three proxy metrics respectively.

As can clearly be observed in the Figure 3.5, there is a strong correlation between each of the proxy metrics and the overall objective. That can be seen as a validation of the use of each and every one of these proxy metrics in place of the overall objective.

With the analysis and rationale from the precedent study as a starting point [7], the work of evaluating and choosing one single proxy metric to center this new set of experiments around began. In the documentation of the study it was recommended that *Minimising average crusher inactivity* should be the proxy metric of choice moving forward [7, p. 25]. Furthermore, as seen in Figure 3.5, this proxy metric displays the most clear correlation to the principal objective. Therefore, *Minimising average crusher inactivity* was chosen as the one and only proxy metric, which will henceforth be referred to as MACI.

3.4.2 Problem instances

As mentioned in Section 3.1.1, six versions of the simulation model were designed, all with slight variations in terms of lengths of road sections and average service rates of shovels. This is to make up a set of problem instances by which to simulate and compare approaches. For the EA experiments all six instances were used, but due to time limitations it was not possible to perform simulations across all six problem instances for the CEA. Three of them were therefore chosen. When choosing three out of the six problem instances, the result files outputted from previous experiments performed by Cox *et al.* [7] were taken into consideration,

and the problem instances that had generated the lowest productivity, thus subject to the largest improvement potential, were chosen. Low productivity seems to be correlated with a large size of the mine. The problem instances that were used in the CEA were number 2, 3 and 6.

3.4.3 Number of trucks

When performing the experiments, the behaviour of the mine was to be examined with different numbers of trucks in use. Each approach was to be examined for the following states of the mine:

- Under-trucked - When there are so few trucks in use that peak productivity cannot possibly be reached. This is never a desirable status quo in a real-life mine, but might be the case temporarily if trucks break down.
- Saturated - When there are as many trucks in use, that peak performance can be reached, but no more. A desired state to be in.
- Over-trucked - When there are more trucks in use than required to achieve peak performance. This might eventually lead to decreased productivity due to more queues and traffic jams.

Since the different problem instances vary in their setup parameters, different numbers of trucks are required to ensure all states are covered. For experiments on each approach and each problem instance examined, ten different numbers of trucks were tested. The different numbers of trucks tested for each problem instance are compiled into table 3.2.

Table 3.2: The different numbers of trucks in use tested during experiments, per problem instance.

No. of trucks	instance 1	instance 2	instance 3	instance 4	instance 5	instance 6
11				x		
12	x			x	x	
13	x		x	x	x	x
14	x	x	x	x	x	x
15	x	x	x	x	x	x
16	x	x	x	x	x	x
17	x	x	x	x	x	x
18	x	x	x	x	x	x
19	x	x	x	x	x	x
20	x	x	x	x	x	x
21	x	x	x		x	x
22		x	x			x
23		x				

3.5 Two phases of the study

Previous work has, as described in Section 2.3, evaluated the use of EAs to generate dispatching schedules and traffic light schedules independently, as well as both of these together using a CEA. This study has been divided into two major phases. Phase one is dedicated to addressing the research question with the approach of using EAs to generate a dispatching schedule, while traffic lights are controlled independently by cyclic timers. This approach will onward be referred to as EA-RT-CL - Evolutionary Algorithm for Real-Time dispatch scheduling with Cyclic Lights. Phase two further develops the work done in phase one by **using an EA to develop the traffic light schedules as well**. Thus, phase two evaluates our research question using the approach of a CEA to generate both dispatching and traffic light schedules. This approach will onward be referred to as CEA-RTL - CEA for Real-time scheduling for Trucks and Lights.

As described in Section 3.3, the general idea is to make use of the solutions in the last generation of the previous EA/CEA run when constructing generation zero of the next run. This can be done in many different ways and this section aims to provide an overview of the approaches that were used in this project.

3.5.1 Phase 1: EA-RT-CL

As mentioned above, the EA-RT-CL approach uses a single EA for the truck dispatching schedules and cyclic timers for the traffic lights. In this phase, three different ways of implementing the use of previous generations were tried. The details of each one are explained below.

Approach 1.1

In approach 1.1 each of the relevant portions of the matching solutions are kept and get a number of different suffixes each. For instance, if there are 25 matching solutions and a population size of 100, each saved "mid-section" would occur four times with four different suffixes, as shown in Figure 3.6. By doing so the hope is to avoid a situation where a randomly generated suffix that happened to be "bad" ruins the chances for an otherwise good solution to survive past the first generation.

Approach 1.2

Approach 1.2 instead adds *one* suffix to each of the matching middle-sections and uses the resulting group of solutions (assuming there are less than 100 matches) to generate the rest of the solutions needed to make up a population size of 100. Using the same example as above; if there are 25 matching solutions, each of them will get one random suffix and the 25 would then be the parents of 75 offspring. The 25 matching solutions and the 75 offspring of the matching solutions together make up generation zero.

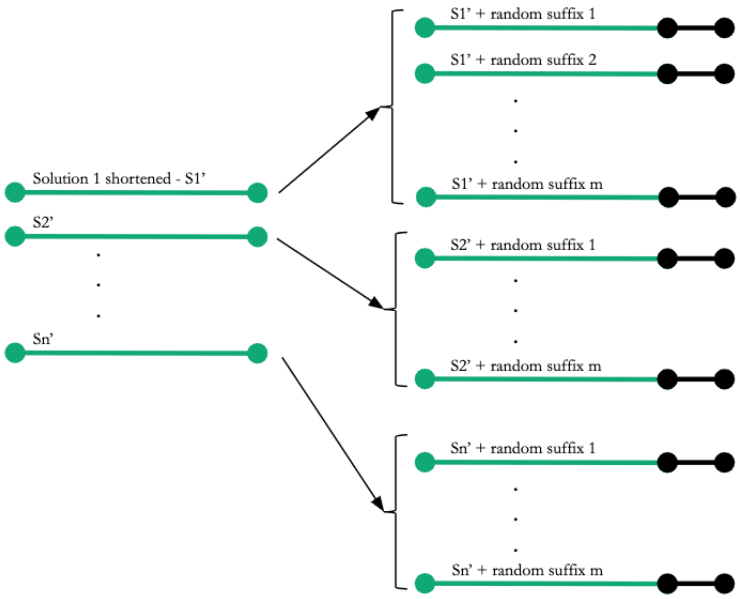


Figure 3.6: Approach 1.1: Adding different suffixes.

Approach 1.3

Approach 1.3 uses the same technique as in 1.1 with the difference that there is a limit put in place, a maximum of the 40 best solutions are kept and used. If the number of matches are fewer than 40, 1.3 will yield the same result as approach 1.1. If there are more than 40 matches, the 40 best ones will be kept and fitted with a number of different suffixes, and the rest will be discarded. Initial experiments showed that 1.3 would yield a different generation zero compared to what 1.1 would have in approximately 40 percent of cases, see Figure 3.7. The underlying data is gathered from pre-experiments for one problem instance only and can therefore only be considered an indicator.

From pre-experiments to experiments

Before running the rather time consuming simulations some pre-experiments were conducted to potentially rule out any approach that did not indicate significant results. It was found that 1.3 displayed the most promising result compared to the original approach. Because of the fact that 1.3 is a direct modification of 1.1, 1.1 was also found to be an interesting path moving along with. Approach 1.2 however, did not show any significant results, either good or bad compared to the other two, why this approach was discarded before large scale experiments.

3.5.2 Phase 2: CEA-RTL

In the second phase of the project changes to the CEA-RTL were implemented, evolving both truck dispatching schedules and traffic light schedules in parallel as explained in Section 3.2.2. Just as in phase 1, three different approaches were tried in the pre-experiments. Having seen satisfactory results in phase 1, all three approaches were designed to be similar to the best

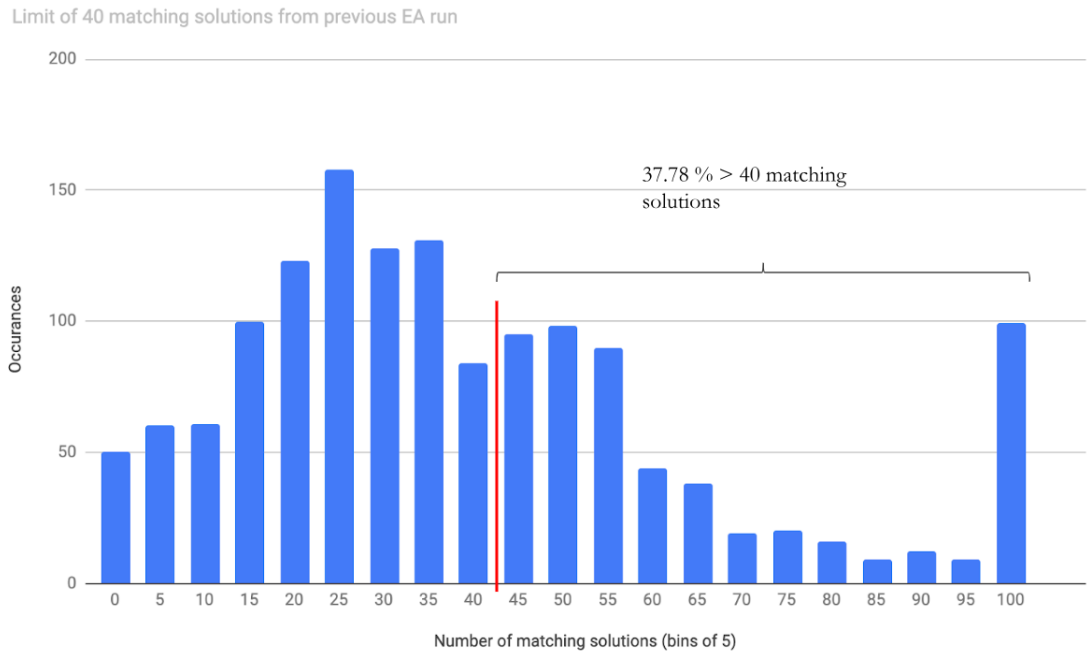


Figure 3.7: A histogram of the number of solutions matching the chosen one in the last generation of a EA run. This number exceeds 40 in approximately 40 percent of the cases.

one from the first phase, approach 1.3. The aim was to study the effect of applying the same concept as approach 1.3 to both populations.

Approach 2.3.1

2.3.1 is very similar to approach 1.3 from phase one. Here, the initial truck dispatching schedules are generated the same way as in 1.3; by saving the 40 best (or less) matches and adding different suffixes to them until the population size of generation zero is 100. The traffic light schedules are generated randomly, just as in the original approach.

Approach 2.3.2

2.3.2 builds directly on 2.3.1 with the difference that the same approach is used to generate the generation zero of the traffic light schedules as well. Constructing the generation zero is done in the same way as earlier explained; by keeping the 40 best (or less) matches and adding different suffixes until the population size is 100.

Approach 2.4

In approach 2.4 the truck dispatching schedules are again generated by keeping the 40 best (or less) matches and adding different suffixes until the population size is 100. The traffic light schedules are generated by keeping the 40 best solutions from last run (best fitness), regardless of whether they match the schedule that was taken into use. The solutions are still re-fitted in time, removing the first part and adding a suffix at the end.

From pre-experiments to experiments

In the pre-experiments it was found that 2.3.2 performed significantly worse and that its performance was unstable (large standard deviation), compared to the other two approaches. Therefore, it was decided that 2.3.2 would be discontinued after pre-experiments.

Chapter 4
Evaluation

In this chapter some of the practical methodology around the experiment setup is briefly described followed by the results of the experiments and related analysis. Throughout the chapter, problem instance 2 is the one being showcased alongside the average of all six problem instances for the EA and the three problem instances (2, 3 and 6) for the CEA. The problem instances display very similar results, and either one could just as well have been chosen. For the complete results, see Appendix A.

The results and analysis are based on the research question:

Can information from previous runs of an evolutionary algorithm be used when creating generation zero in the following run of the evolutionary algorithm in order to improve its performance?

The phrase "Improving performance" can be interpreted as either one of the following three:

1. To increase overall productivity - An increased level of peak performance, i.e. maximised productivity in terms of number of truckloads being delivered to the crusher during a shift.

2. To achieve better productivity using a smaller number of trucks - Reaching a higher number of truckloads for cases when the number of trucks is lower than in a saturated state of the mine (minimum number of trucks in order to reach 99% of peak performance).

3. Less computational power required to run the evolutionary algorithm - A smaller number of generations required to be run through by the evolutionary algorithm to find a solution, and thus less computational power and time required.

After developing and implementing the changes to the algorithm (see Section 3.5), experiments were run in the simulator. The simulator and the algorithms to be evaluated were programmed in Java, with the extension of the external library *lpsolve* to solve linear programming problems [4].

The simulation runs were set up as follows:

1. One full shift simulation represents a work shift of 500 minutes with a fixed number of trucks
2. Ten different number of trucks for each approach (original + experimental approaches) were evaluated
3. Six and three different problem instances for the EA and CEA respectively, with slightly different set-ups of the mine simulator (see Section 3.1.1), were run
4. 25 runs of each simulation setup were run. Results computed as the average of the output for all 25 simulations

The simulations were run on multiple desktop computers in parallel, with as many as 30 running computers with six processes on each on occasions. The results of each approach were compared to those of the original approach. They were all run during night time, spread out on approximately four weeks.

4.1 Evolutionary algorithm

The results that were observed in Phase 1: EA-RT-CL are shown in tables and graphs below. The two main areas that were examined were (1) productivity (number of truckloads) and (2) number of generations.

4.1.1 Mine productivity

Being the overall objective of the mine, the productivity in terms of number of truckloads is central when talking about performance of the EA. Table 4.1 and Figure 4.1 display the productivity, i.e. number of truckloads by different number of trucks for problem instance 2. Both approach 1.1 and 1.3 perform similarly compared to the original approach.

The largest difference can be seen for 15 trucks, where original is outperformed by 1.1 with 1.48 truckloads, and 1.3 with 1.28 truckloads. This is per full shift of 500 minutes. On average (across the different numbers of trucks), 1.1 deviates from original with 0.328 truckloads and 1.3 deviates from original with 0.244 truckloads. The average standard deviations (across the different numbers of trucks) for original, 1.1 and 1.3 are 0.535, 0.540 and 0.485 respectively.

Here, problem instance 2 is the one being portrayed, but the same trend can be seen for all problem instances. Figure 4.2 shows the average productivity for all instances.

None of the approaches reach a significantly higher peak performance than the original approach. In an under-trucked state of the mine, primarily around 14-15 trucks, the productivity can be seen to be slightly better than original. However, it does not reach *peak performance* with any fewer trucks than before.

Table 4.1: EA. Problem instance 2. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
14	137.88	0.99	137.44	0.98	-0.32%	138.40	0.75	0.38%
15	143.36	0.74	144.84	0.73	**1.03%**	144.64	0.84	**0.89%**
16	149.56	0.70	149.56	0.80	0.00%	149.44	0.75	-0.08%
17	152.76	0.65	153.08	0.63	0.21%	152.72	0.66	-0.03%
18	153.76	0.43	153.84	0.37	0.05%	153.92	0.39	0.10%
19	154.76	0.51	154.04	0.20	-0.47%	154.84	0.46	0.05%
20	154.92	0.27	155.04	0.45	0.08%	155.04	0.34	0.08%
21	155.04	0.20	155.04	0.34	0.00%	155.04	0.20	0.00%
22	155.00	0.57	154.92	0.39	-0.05%	155.04	0.45	0.03%
23	154.92	0.27	154.88	0.52	-0.03%	155.00	0.00	0.05%
Total	1511.96		1512.68			1514.08		
Average		0.53		0.54			0.48	
Diff. prod.					**0.05%**			**0.14%**

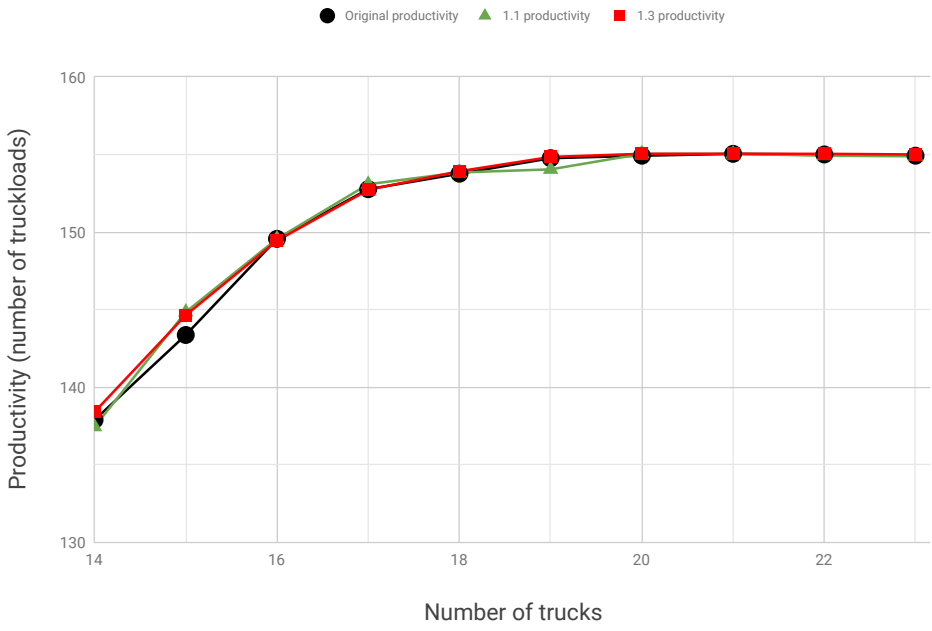


Figure 4.1: EA. Problem instance 2. Average mine productivity by the number of trucks in use.

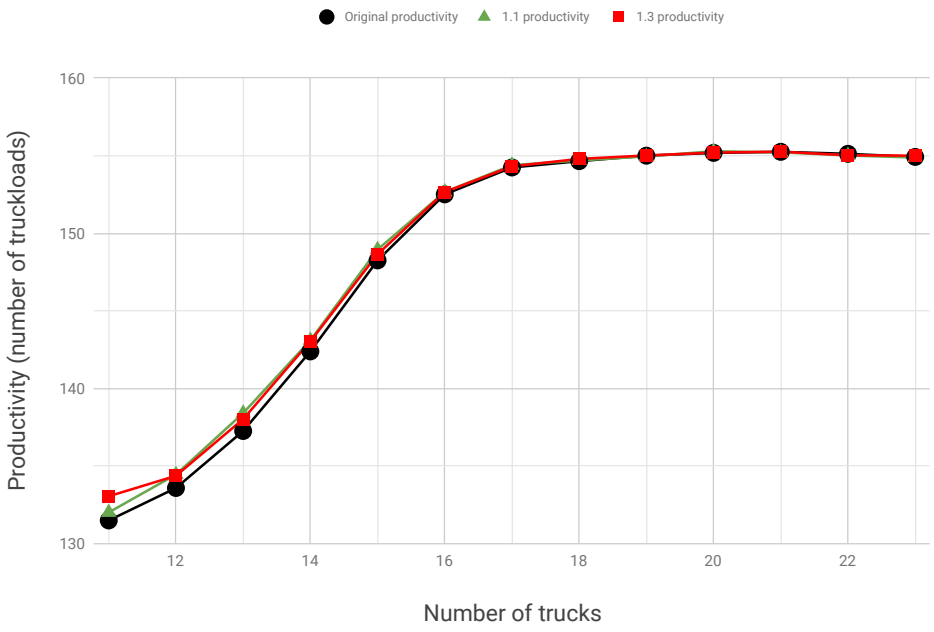


Figure 4.2: EA. Average of all six problem instances. Average mine productivity by the number of trucks in use.

4.1.2 Number of generations

Averages

Table 4.2 and Figures 4.3 and 4.4 show the average number of generations (by number of trucks) that the EA requires to reach a solution (i.e. the number of generations that were run before the EA was terminated by one of the two termination reasons described in Section 3.2.1). Figures 4.3 and 4.4 display problem instance 2 and the average of all six problem instances respectively. Both 1.1 and 1.3 outperform the original approach and require a lower number of generations for all different numbers of trucks in use.

In total (counting the number of generations for all trucks for all six problem instances) approach 1.1 requires 11.65 percent less generations to reach a solution compared to original. For 1.3 the same number is 11.22 percent. The average standard deviation across all problem instances and all trucks are 86.17, 82.68 and 83.69 for original, 1.1 and 1.3 respectively.

Table 4.2: EA. Problem instance 2. Number of generations, standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
14	218.74	54.89	205.36	55.28	-6.12%	211.84	60.01	-3.15%
15	236.50	64.48	217.59	64.80	-8.00%	225.00	66.38	-4.86%
16	252.60	74.27	229.04	72.39	-9.33%	227.26	72.00	-10.03%
17	252.85	80.32	225.34	77.33	-10.88%	219.33	68.77	-13.26%
18	198.94	98.84	164.42	101.69	-17.35%	157.00	102.39	-21.08%
19	154.46	101.01	132.26	103.14	-14.37%	130.63	108.53	-15.43%
20	137.96	111.59	121.46	100.29	-11.96%	125.48	105.89	-9.05%
21	138.46	104.61	123.00	102.10	-11.17%	102.41	98.05	-26.04%
22	102.85	100.45	92.52	101.62	-10.04%	98.78	104.00	-3.95%
23	122.05	106.45	101.96	99.74	-16.46%	101.21	106.82	-17.07%
Total	1815.40		1612.94			1598.94		
Average		89.69		87.84			89.28	
Diff. gens.					**-11.15%**			**-11.92%**

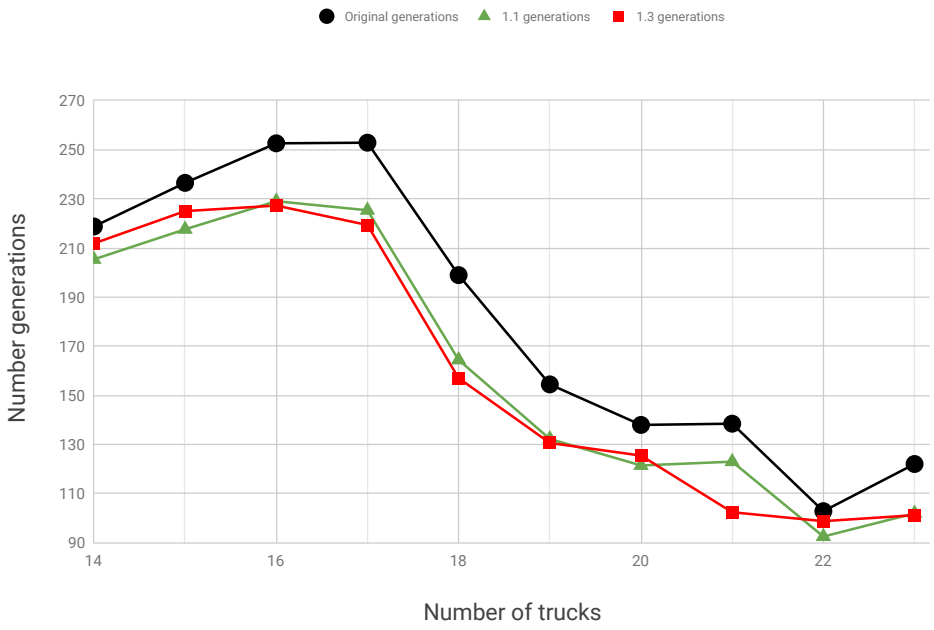


Figure 4.3: EA. Problem instance 2. Average number of generations required by the EA to find a solution, by the number of trucks in use.

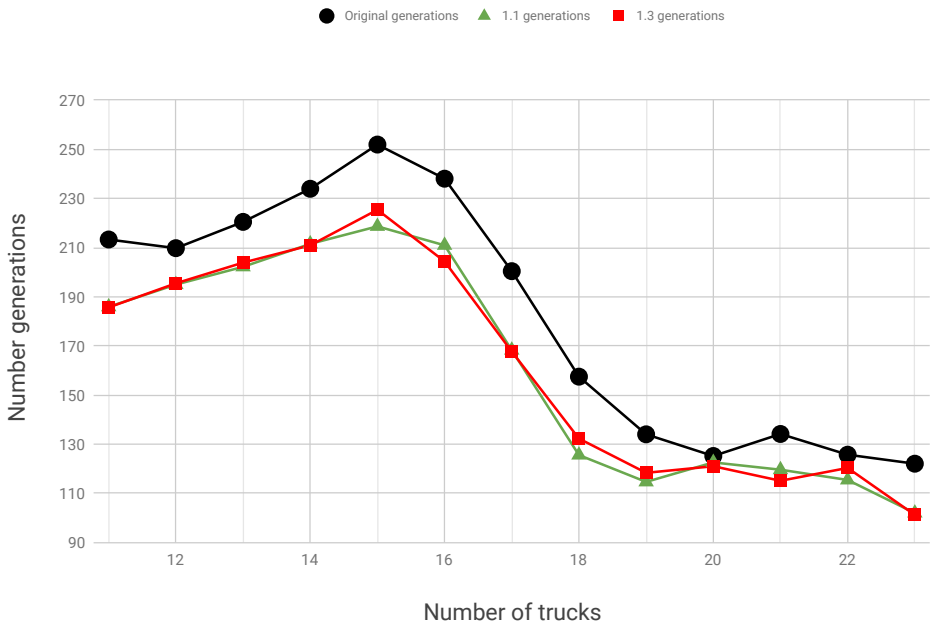


Figure 4.4: EA. Average of all six problem instances. Average number of generations required by the EA to find a solution, by the number of trucks in use.

Distribution

In order to ascertain the witnessed gap between the average number of generations run by ~~the EA for the respective approaches, this was closer examined by being broken down on~~

truck level. Looking at a specific case, Figure 4.5 shows the distribution of the number of generations required to find a solution for 17 trucks in problem instance 2. The normal distribution curve plotted is centered around a higher number for the original approach compared to 1.1 and 1.3. It should be kept in mind that a low number is desirable.

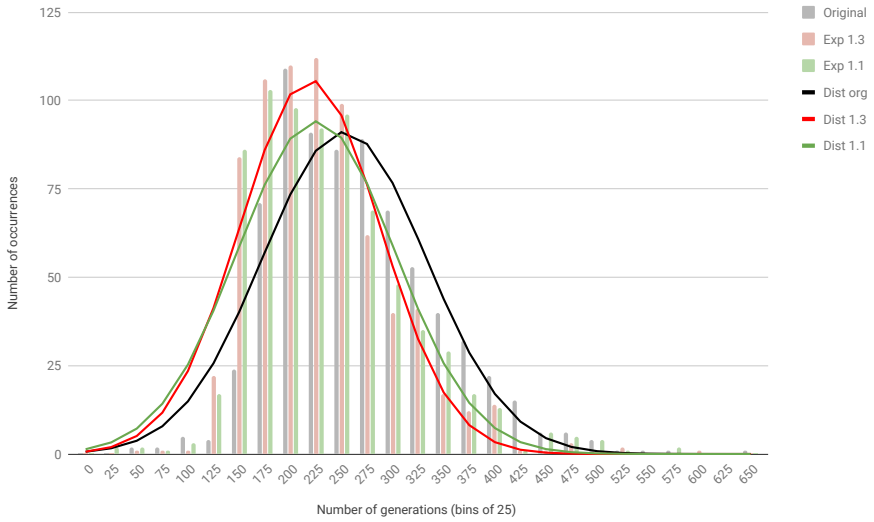


Figure 4.5: EA. Problem instance 2. 17 trucks. Distribution of number of generations required by the EA to find a solution, displayed as histogram and normal distribution curves.

4.1.3 Analysis

The results from the EA were perceived as promising, primarily in regards to the observed reduction in the number of generations required to reach a solution. Worth mentioning is also the improvement in productivity for the under-trucked systems, as could be seen in Figures 4.1 and 4.2.

On this basis, it seems increased performance cannot be seen in the form of the first "sub-meaning" of the research question; increasing the peak performance in terms of number of truckloads. Looking at the second however, an improvement could be observed as productivity increased in under-trucked systems. The *peak* performance still cannot be reached with fewer trucks and a real life mine presumably seldom operates at such an under-trucked state as where the improvement was seen, but the result is still perceived as interesting.

Looking at the number of generations, good results were observed. Both 1.1 and 1.3 show a remarkable difference compared to original. "Sub-meaning" number three in regards to performance improvement, decreasing the required computational power and time, can thus be considered achieved. The correlation between computational power, number of generations and run time is further investigated in Section 4.3.

Standard deviation

As with all experiments, results need to be statistically proven with some degree of certainty, in order to claim a definitive difference between approaches. One wants to be absolutely sure that observed behaviours are not a product of coincidence. Some of the results produced in this study show large deviations between simulations, leading to large confidence intervals and thus they cannot be interpreted as absolute truths.

The most prominent result from the study is the observed difference in number of generations required to find solutions. As commented on in Section 4.1.3, a rather large difference can be seen (a total decrease of 11.22 percent with approach 1.3 compared to original over all problem instances). This however, is a comparison between the average numbers produced by 25 full shift simulations. Figure 4.6 depicts the average number of generations for problem instance 2 - same as in 4.3 with the addition of a 95 percent confidence interval (dotted lines) for the EA (original and 1.3). Here, approach 1.3 is the only one plotted alongside original since it displayed the largest difference, and is thus the most likely to avoid much overlapping confidence intervals. However, this was not the case. The confidence intervals overlap a lot and it can therefore not be guaranteed that the observed differences between the approaches are not coincidental.

The more overlap of two confidence intervals (representing two independent groups), the less likely it is for these two groups to be distinguishable by a certain variable. For reference, when comparing two independent groups, 95 % confidence intervals that display a proportion overlap of 0.5 or less correspond to $p < 0.05$ [10]. In the same way, 95 % confidence intervals that do not overlap but just touch mean $p < 0.01$. The proportion overlap is the overlap in absolute numbers divided by the average margin of error for the two groups. These relationships are sufficiently accurate when both sample sizes are at least 10, and the margins of error do not differ by more than a factor of 2. Nonetheless, the confidence intervals in this case are overlapping much more, rejecting statistical significance by any degree.

In order to statistically affirm that the results are valid, a lot more experiments would have to be run (if possible at all). This would then ideally strengthen the validity of the conclusion, but would presumably not lead to any new findings. Nevertheless, given the nature of this area of science and especially the industry, the results do not need to be scientifically proven in order to be perceived as interesting and potentially valuable.

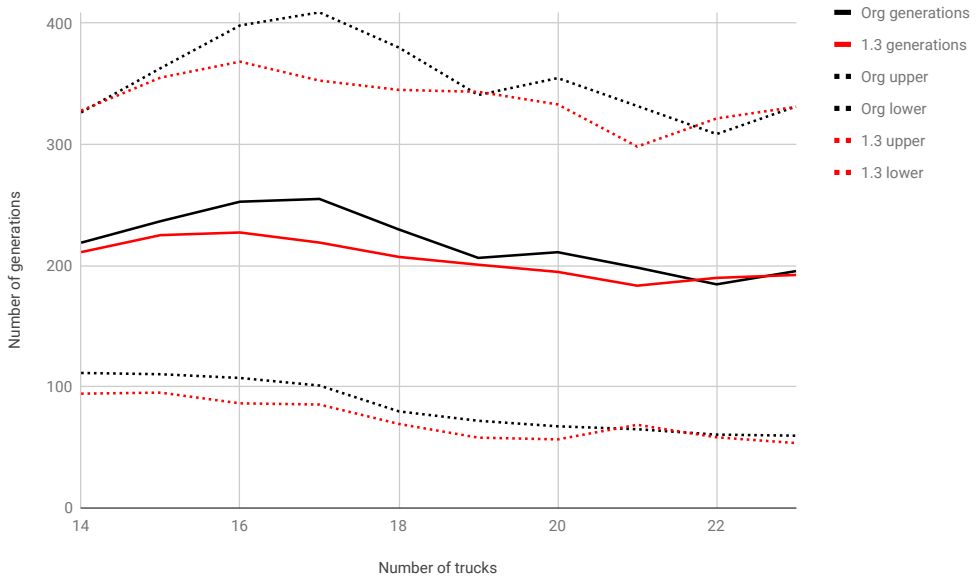


Figure 4.6: EA. Problem instance 2. Average number of generations required by the EA to find the solutions, with a 95% confidence interval.

4.2 Co-evolutionary algorithm

The results that were observed in Phase 2: CEA-RTL are shown in tables and graphs below. The two main areas that were examined were (1) productivity (number of truckloads) and (2) number of generations.

4.2.1 Mine productivity

In regards to mine productivity, i.e. number of trucks being delivered to the crusher during a 500 minute shift and subsequently the overall goal, interesting results were produced by the CEA approaches. Opposed to what was observed in results from phase 1, a more significant difference between the approaches was detected. The results from approach 2.4 proved to be consistently lower than those of the other two approaches. The original approach and 2.3.1 performed very similarly. Table 4.3 and Figure 4.7 demonstrate exactly this for problem instance 2. These findings proved pervading throughout all three instances, as can be interpreted from Figure 4.8 which displays the average across the three instances 2, 3 and 6.

The largest difference that can be seen in Figure 4.7 is for 15 trucks, where original performs similarly to 2.3.1 and better than 2.4. The difference between original and 2.4 is here 2.16 truckloads. On average, for problem instance 2, 2.3.1 deviates from original by a decrease of 0.096 truckloads and 2.4 deviates from original by a decrease of 0.680 truckloads (across the different number of trucks). This can be compared to the average standard deviations for original, 2.3.1 and 2.4, which are 0.481, 0.475 and 1.104 respectively.

Table 4.3: CEA. Problem instance 2. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
14	140.80	0.5800	140.76	0.66	-0.03%	139.48	1.2600	-0.94%
15	147.08	0.57	147.20	0.71	0.08%	145.04	**4.6000**	-1.39%
16	151.52	0.77	151.48	0.59	-0.03%	150.96	0.9300	-0.37%
17	154.00	0.29	153.96	0.54	-0.03%	153.48	0.5900	-0.34%
18	154.52	0.51	154.64	0.49	0.08%	154.36	0.6400	-0.10%
19	154.84	0.37	155.00	0.41	0.10%	154.56	0.5800	-0.18%
20	155.00	0.41	155.16	0.47	0.10%	154.64	0.5700	-0.23%
21	155.08	0.4000	155.04	0.35	-0.03%	154.64	0.6400	-0.28%
22	155.20	0.41	155.04	0.20	-0.10%	154.80	0.4100	-0.26%
23	155.20	0.5	155.12	0.33	-0.05%	154.48	0.8200	-0.46%
Total	1523.24		1523.40			1516.44		
Average		0.48		0.48			1.10	
Diff. prod.					**0.01%**			**-0.45%**

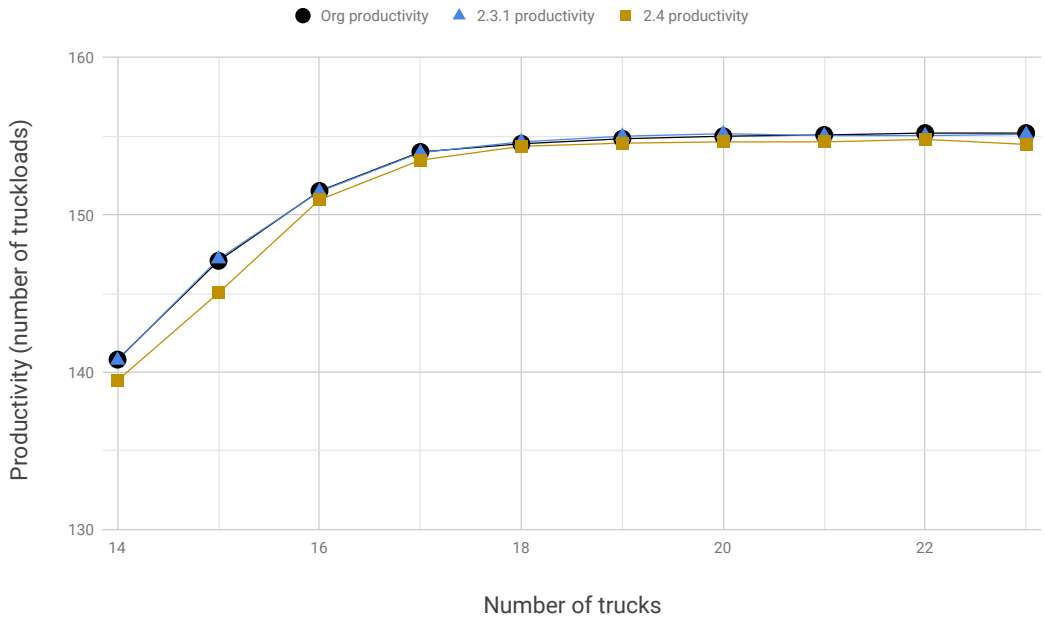


Figure 4.7: CEA. Problem instance 2. Average mine productivity by the number of trucks in use.

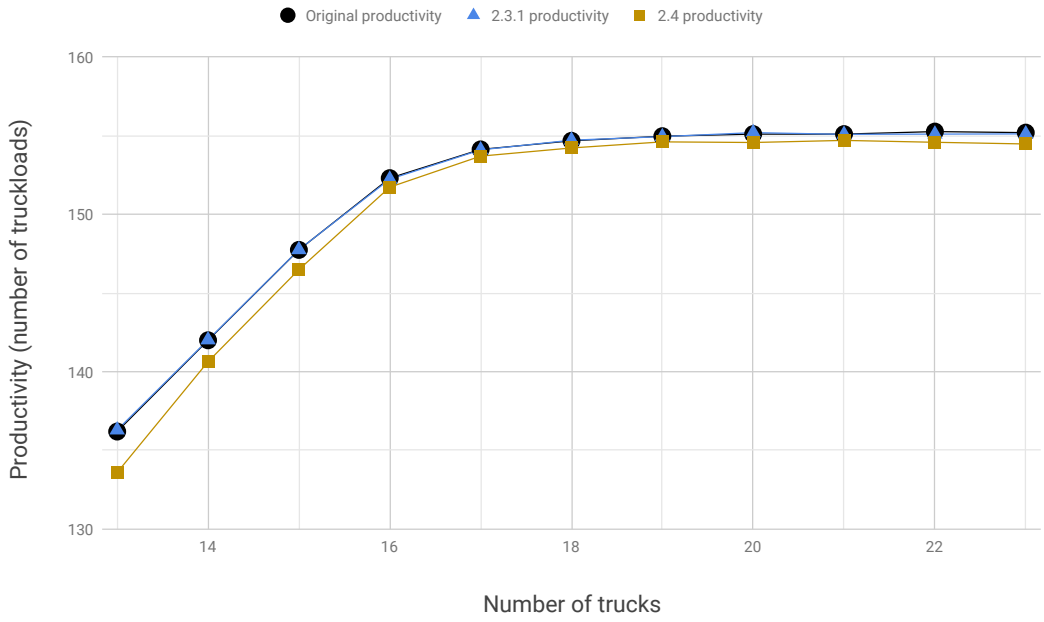


Figure 4.8: CEA. Average of problem instances 2, 3 and 6. Average mine productivity by the number of trucks in use.

4.2.2 Number of generations

Averages

In phase 1, findings were seen regarding the number of generations required by the EA to find a solution. For the CEA however, these results demonstrated an even bigger difference between the approaches. The original approach and 2.3.1 perform very similarly, whereas 2.4 requires a lot less generations. This is depicted in Table 4.4 and Figure 4.9, where results for problem instance 2 are plotted. These findings were seen across all three instances, which Figure 4.10 (compiling the average of all instances) testifies to.

In total (counting the number of generations for all trucks for problem instance 2, 3 and 6) approach 2.3.1 proves no significant difference; a 0.26 percent increase compared to original. For approach 2.4, a **33.84 percent decrease** is the corresponding number. N.b. these numbers differ from the ones in the bottom of Table 4.4 because this is computed from three instances together, as opposed to Table 4.4 which only displays problem instance 2. The average standard deviation across all problem instances and all trucks are 127.24, 128.59 and 115.81 for original, 2.3.1 and 2.4 respectively.

Table 4.4: CEA. Problem instance 2. Number of generations, standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
14	385.15	91.74	381.35	101.05	-0.99%	268.64	87.71	-30.25%
15	403.09	105.53	403.37	103.26	0.07%	282.42	93.81	-29.94%
16	417.94	115.92	418.49	116.33	0.13%	290.63	92.45	-30.46%
17	395.42	115.07	395.93	131.28	0.13%	257.18	110.41	-34.96%
18	337.53	125.75	327.92	126.91	-2.85%	198.71	126.05	-41.13%
19	275.04	142.80	264.83	147.37	-3.71%	153.50	139.26	-44.19%
20	222.41	144.09	224.92	135.58	1.13%	129.03	137.82	-41.99%
21	231.81	142.21	227.32	141.27	-1.94%	100.74	129.60	-56.54%
22	232.70	136.54	222.06	146.15	-4.57%	114.56	130.32	-50.77%
23	237.70	137.41	223.34	141.08	-6.04%	127.30	137.61	-46.45%
Total	3138.79		3089.53			1922.71		
Average		125.71		129.028			118.504	
Diff. gens.					**-1.57%**			**-38.74%**

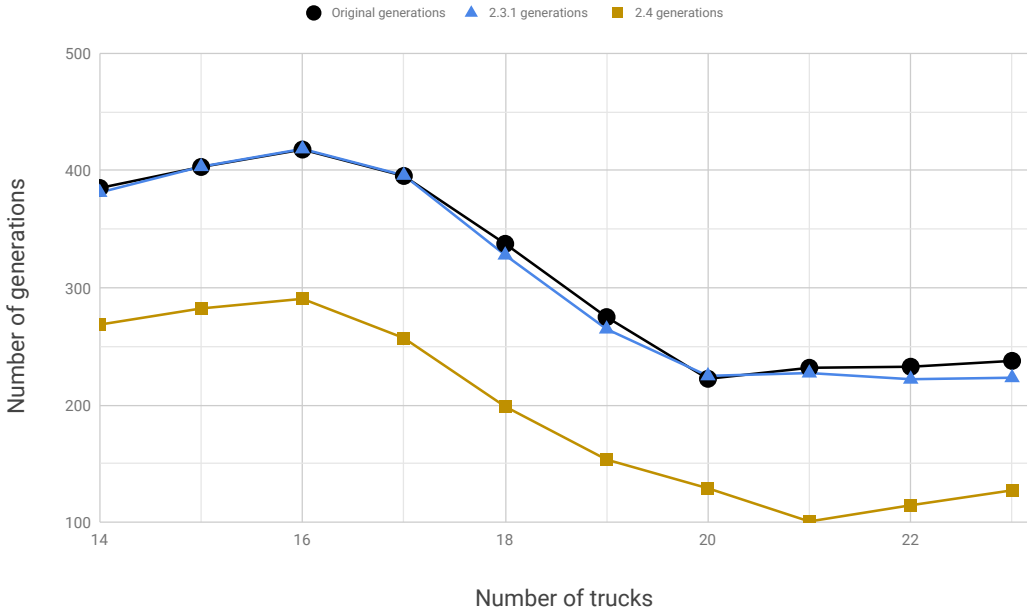


Figure 4.9: CEA. Problem instance 2. Average number of generations required by the CEA to find a solution, by the number of trucks in use.

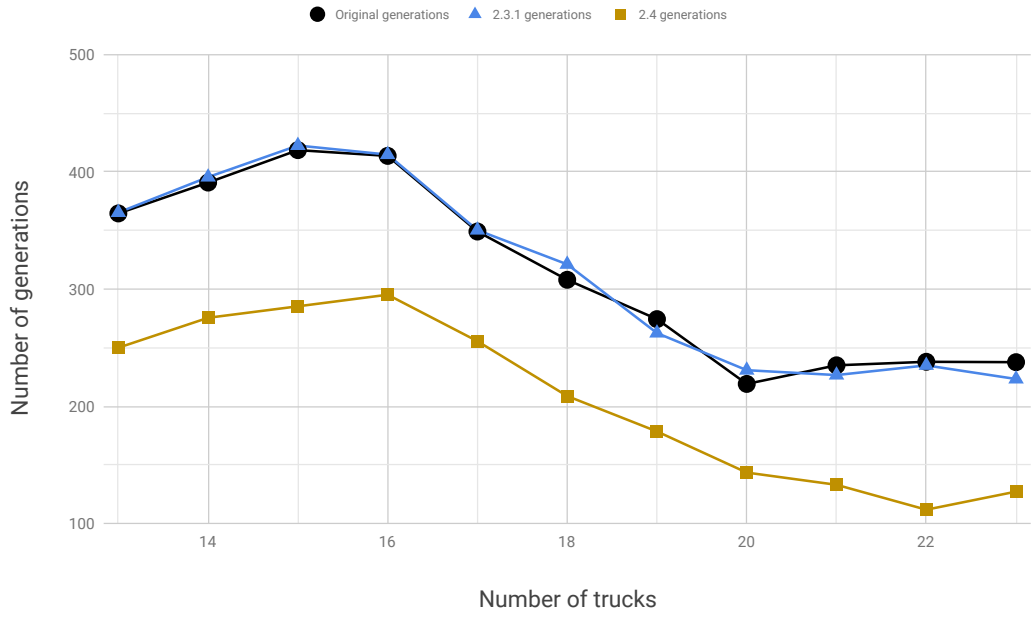


Figure 4.10: CEA. Average of problem instances 2, 3 and 6. Average number of generations required by the CEA to find a solution, by the number of trucks in use.

Distribution

Again, looking at a specific case, Figure 4.11 shows the distribution of the number of generations required to find a solution for 17 trucks in problem instance 2. The normal distribution

curve plotted is centered around a lower number for approach 2.4 compared to the other two.

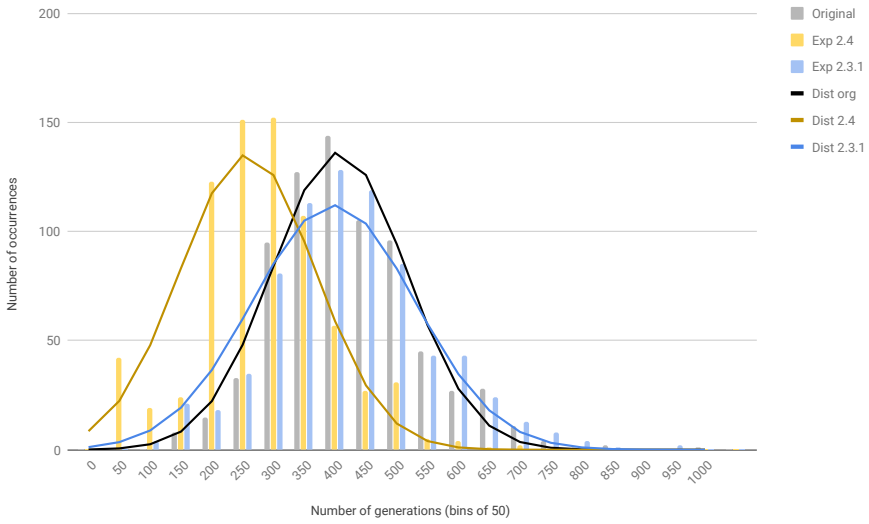


Figure 4.11: CEA. Problem instance 2. 17 trucks. Distribution of number of generations required by the CEA to find a solution, displayed as histogram and normal distribution curves.

4.2.3 Analysis

Many of the implemented changes were similar for the EA and the CEA as explained in Section 3.5. Due to this, it was expected that similar results were to be seen for the CEA as was for the EA. Looking into the results of the CEA however, these deviate somewhat from the pattern seen in Phase 1.

Starting with the mine productivity, it was seen that changes made to the algorithm actually had the potential to make the productivity worse. Being the central objective in any mine business, this is of course highly undesirable.

Approach 2.3.1 only used the previous *truck dispatching* populations and produced the generation zero traffic light schedules the same way as in the original approach (randomly). This resulted in the productivity being roughly the same as for the original approach, thus so far in line with the results from the EA approach 1.3. However, the improvement seen in number of generations for the EA approach 1.3 could not be seen for the CEA. The number of generations required to reach a solution for approach 2.3.1 was very similar to the original approach. Hence, no real improvement could be witnessed at all, in spite of implementing a change very similar to the one that produced good results for the simpler EA.

One theory to explain this behaviour is that when the traffic light schedules are still generated at random (in the beginning of each CEA run) the time it takes for them to evolve until ceased improvement will be the same as in the original approach. Termination of the CEA run will only happen when *both* populations have stopped improving, thus implicating that the changes have to be implemented to both populations in order for any real change in results to occur.

A suggestion for further investigation is to explore what would happen if the opposite was tried; producing generation zero truck dispatching schedules randomly and traffic light schedules using the previous solutions. An initial hypothesis is that this might generate somewhat of a result, opposing what was seen for approach 2.3.1. The reasoning behind this is the following: Looking at the different solution representations, the truck dispatching schedules are limited in terms of what can happen as each position in the solution can only be one out of four options (the four shovel IDs). The traffic light schedules on the other hand, have a lot more room for variation, as the "green times" are not limited to any number of options. The "green times" (one on each position in the traffic light schedule) can be any number up to a high set limit. When the traffic light schedules are generated randomly (as in approach 2.3.1), they can potentially get relatively much *worse* compared to what truck dispatching schedules can ever become. Consequently, this can put the CEA in a relatively bad starting point, hence requiring to run through more generations until finding satisfactory solutions. Generating the truck dispatching schedules randomly and using previous populations for the generation zero traffic light schedules will, with this reasoning, never put the CEA in such a bad place as a starting point. The CEA might thus reach a good solution quicker.

In approach 2.4 changes were implemented for both schedule populations, although of a different character. Because of the good results seen in 1.3 the generation zero truck dispatching schedules are again developed with the same technique. For the traffic light schedules on the other hand, the best schedules from the previous CEA run are used, regardless of whether or not they match the schedule that was last used. This proved to lower the required number of generations a lot. This fact somewhat reinforces the reasoning above; the traffic light schedules will need a lot of time to get to the desired quality, but when the best schedules from the previous generations are used, the algorithm can begin evolving the schedules from a better starting point. This enables them to reach the desired quality faster since most of the "crazy" suggestions of timings have been discarded already.

It is noteworthy however, that the productivity seems to suffer from this approach, showing a small general decrease and a somewhat unstable performance with a couple of remarkable outliers.

One potential explanation to this behaviour lies in the implementation of the construction of the generation zero traffic light schedules. When initially generating the schedules, they are usually made slightly longer than required, to ensure always being of at least sufficient length. The simulations performed to evaluate fitness, however, will only evaluate over the time H (approximately 60 minutes), potentially leaving the last part of the schedule unevaluated. In the original approach this does not affect the outcome since (1) the last part of the schedule never gets used (there is a schedule update before that) and (2) all schedules are discarded and new ones are generated randomly at the beginning of the next EA run. Some of the changes implemented as a part of this study, primarily the re-scaling to fit time (explained in Section 3.3), will however take the very last part of the schedule into use. "Moving back" the time frame by cutting off the first part will result in the un-evaluated part becoming included in the evaluation simulations. In some of these cases this last part of traffic light times will be particularly off, causing the simulation to produce worse results.

Now, if bad solutions are produced when all schedules are generated at random, these solutions would be discarded at an early stage, and the algorithm would move on to search in other parts of the search space. The problematic difference with the changes implemented in 2.4 is that the same middle sections occur many times with different suffixes. No completely

new solutions are produced, potentially causing the CEA to get stuck in a narrow search space that does not produce good solutions. Combining these two approaches and creating a generation zero that consists of some random solutions and some that are kept from last run the way they are in 2.4, could potentially produce a better result. This course of action is hoped to mitigate the risk of narrowing the search space too much, at the same time as keeping and utilising qualitative information. This is one idea for future work in the area.

Standard Deviation

Just as with the EA, in order to statistically affirm any results the confidence intervals would need to be completely separate from each other. Again, the most significant difference observed is the number of generations for approach 2.4 compared to original, the total decrease over problem instances 2, 3 and 6 being 33.84 percent. Figure 4.12 displays the same average numbers for the CEA (original and 2.4) for problem instance 2 as Figure 4.9, with the addition of a 95 percent confidence interval (dotted lines). Despite the large difference between the two curves, the confidence intervals overlap and thus it cannot be confirmed that the difference showed in the experiments are not coincidental. As mentioned in Section 4.1.3, in order to validate the results a lot more simulations would have to be run if possible at all. Even so, these results, just as the results produced by Phase 1, are considered interesting and valuable given the nature of this area of science.

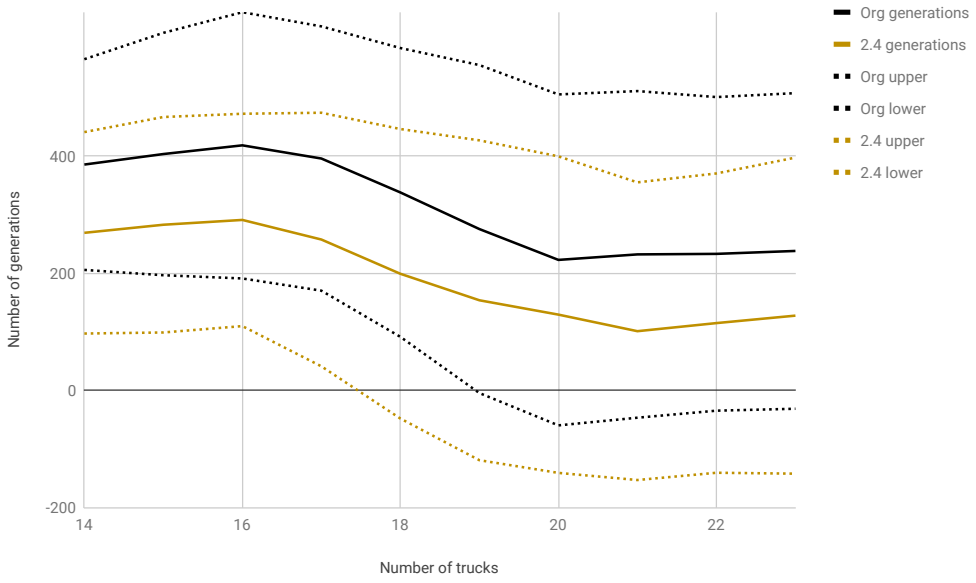


Figure 4.12: CEA. Problem instance 2. Average number of generations required by the CEA to find the solutions, with a 95% confidence interval.

Instability in productivity for approach 2.4

Looking more closely at the mine productivity of the different approaches, the same conclusions, or lack thereof, apply. It cannot be stated for sure that the results in the graphs depict

an absolute truth since the confidence intervals overlap.

Moreover, looking at the standard deviation for approach 2.4 in Figure 4.14, the lines plotted here are wider apart than the ones belonging to the other approaches in the same graph. For 15 trucks this gap is remarkably large, the underlying reason being two notable outliers, see Table 4.5. These outliers are also visualized in Figure 4.15, where the variation is displayed clearly. What this tells us is that 2.4 is potentially more unstable in terms of mine productivity. It is more prone to produce outliers. This is likely due to the phenomenon elaborated on in Section 4.2.3, where including un-evaluated parts of previous solutions can lead to worse results because of a narrow and disadvantageous search space.

When examining the other problem instances for approach 2.4, scrutinizing also their output for outliers, it was seen that this was never as severe a case for either problem instance 3 or 6. The results where slightly lower and more unstable than for approach 2.3.1 and original, with occasional samples that could almost be classified as outliers, but the extreme drop in productivity was never seen. Visualisations of this can be observed in Appendix A.2.2. Nonetheless, even as only two out of 75 full shift simulations produced exceptionally low output, that is reason for not moving forward with the CEA approach 2.4 until root causes are identified and a safe and stable output can be expected.

Comparing the results for the CEA with the ones for the EA, it can be seen (Figure 4.13) that the productivity for the EA is more stable, i.e. the confidence interval is more narrow.

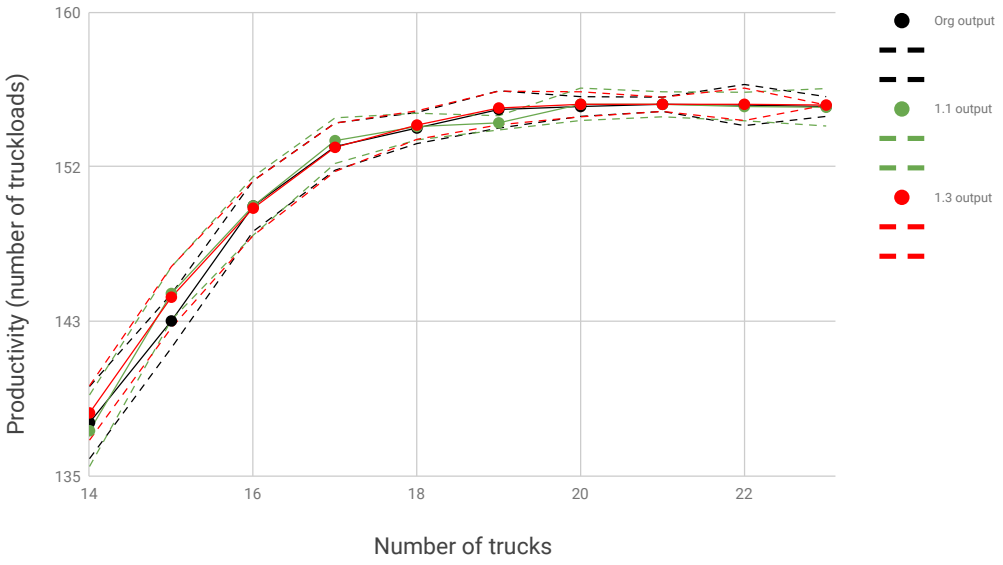


Figure 4.13: EA. Problem instance 2. Average number of truckloads being delivered during a shift, with a 95% confidence interval.

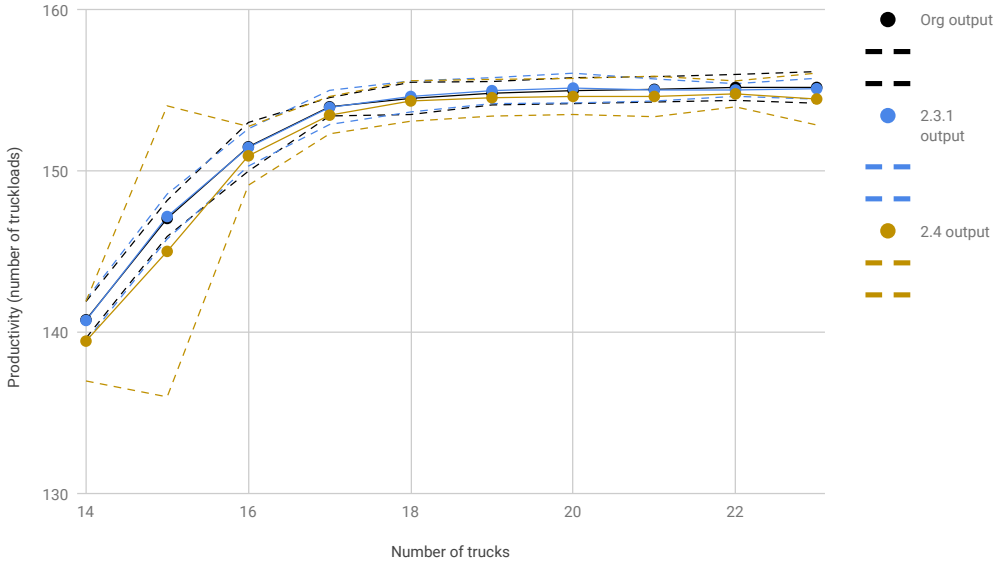


Figure 4.14: CEA. Problem instance 2. Average number of truckloads being delivered during a shift, with a 95% confidence interval.

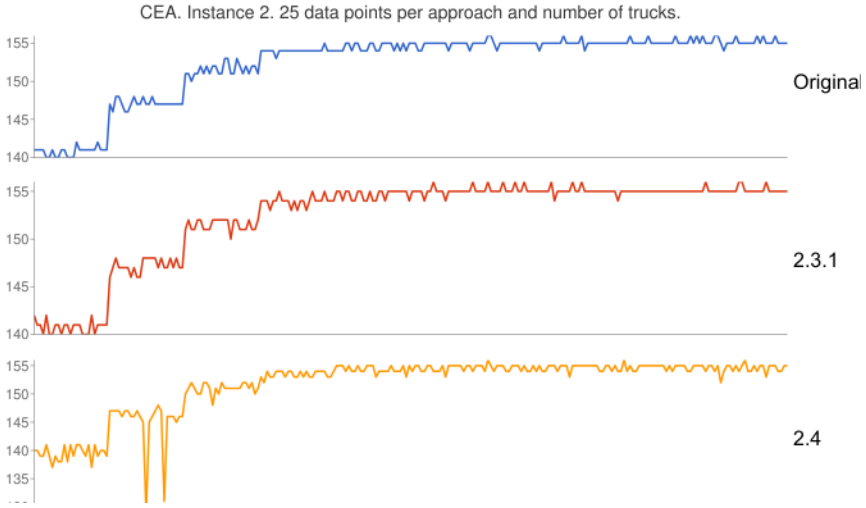


Figure 4.15: CEA. Problem instance 2. Number of truckloads by approach and number of trucks in use (25 samples per number of trucks). Highlights the two outliers produced by approach 2.4 for 15 trucks.

Table 4.5: Approach 2.4. Problem instance 2. 15 trucks. Productivity (number of truckloads).

No. simulation	Productivity
1	147
2	147
3	147
4	147
5	146
6	147
7	147
8	146
9	146
10	147
11	146
12	145
13	**129**
14	145
15	146
16	147
17	148
18	147
19	**131**
20	146
21	146
22	146
23	145
24	146
25	146

4.3 Run time and number of generations

Decreasing the number of generations is presumably closely related to limiting the required computational power, as it means fewer calculations will be required. Reducing the required computational power is of little use if it does not mean that calculations can be completed in a shorter amount of time. In the following section, the notion of assuming a correlation between the number of generations and computational power and run time is investigated.

The primary metric used when measuring the required computational power for the different setups was the number of EA or CEA generations before termination of the run. Number of generations has the benefit of being a very exact way of measuring, as the output is an exact number for each run. However, the overall value that we want to achieve with decreased computational power is decreased run time. Measuring the run time directly is difficult due to the fact that it is generally dependent on an array of other factors. For instance, it will vary depending on the type of computer used and the amount of processes running in parallel. Efforts were made to mitigate these factors in order to also produce time-based results that could be analysed. Most of the experiments were run during night time and on the same desktop computers across a time period of approximately four weeks. Most of the time six processes were running in parallel, with some exceptions. Although not exact, the time-based results were collected and compared to the number of generations to confirm that they were in fact correlated.

When plotting the average number of generations alongside the average run time for a full shift (both by number of trucks), it can be seen that the two graphs display a very similar shape and follow one another another closely. Figure 4.16 and 4.17 depict this for the single EA. However, what is stated above is not true for 14 and 15 trucks. This indicates that one should not trust the connection between generations and run time blindly, even though it is consistently present in a large majority of the cases. On the other hand, these exceptions should not be cause of too much concern, as they may be a result of the error sources described above. The corresponding numbers (number of generations and run times for a full shift by number of trucks) for the CEA, can be found consolidated into Figure 4.18. Here, the above described reasoning is followed consistently.

In figure 4.19 the average number of generations and the average run time per run of the CEA is plotted. The link between these two can be seen as more relevant since it in the end is the run time for a EA/CEA run that is to be minimised. The run time for the full 500-minute shift includes start-up times etc. that would not exist in real mining operations. Therefore, it is reassuring that the correlation between number of generations and run time per CEA run is present as well.

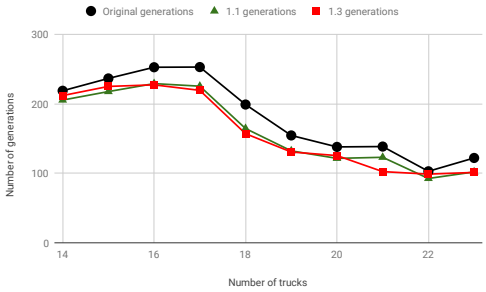


Figure 4.16: EA. Problem instance 2. Average number of generations per number of trucks in use.

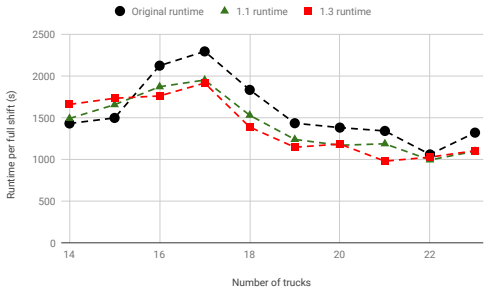


Figure 4.17: EA. Problem instance 2. Average run time for a full shift (500 min) per number of trucks in use.

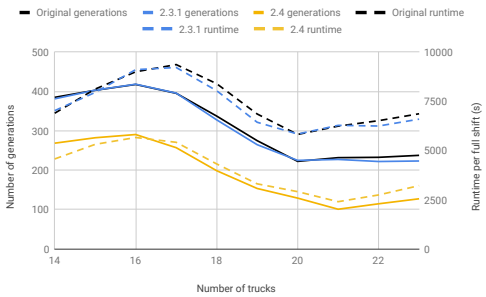


Figure 4.18: CEA. Problem instance 2. Average number of generations and average run time for a full shift (500 minutes) per number of trucks in use.

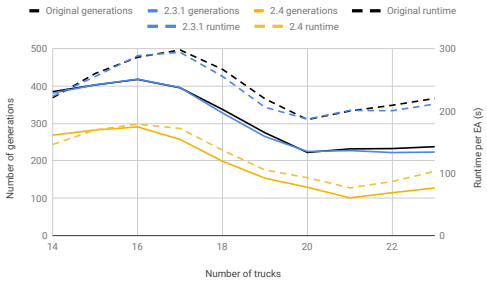


Figure 4.19: CEA. Problem instance 2. Average number of generations and average run time for a single CEA run per number of trucks in use.

Figures 4.20 and 4.21 depict the correlations between the number of generations (the average number of generations of all EA runs during one simulation) and the run time for the corresponding simulation. Figures 4.22 and 4.23 depict the corresponding for the CEA. As can be seen, the two are closely related and a clear correlation can be observed. This confirms that although run time is the more interesting metric in this context, number of generations can be trusted as an indicator of time and computational power.

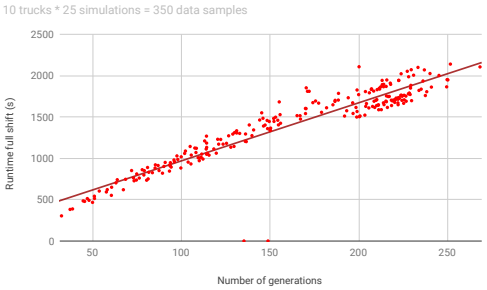


Figure 4.20: EA. Problem instance 2. Approach 1.3. Correlation between number of generations and run time.

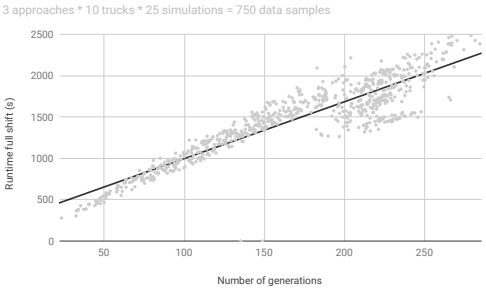


Figure 4.21: EA. Problem instance 2. All approaches together. Correlation between number of generations and run time.

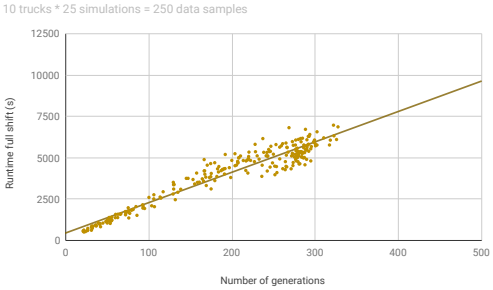


Figure 4.22: CEA. Problem instance 2. Approach 2.4. Correlation between number of generations and run time.

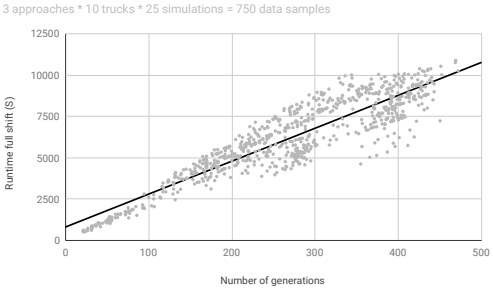


Figure 4.23: CEA. Problem instance 2. All approaches together. Correlation between number of generations and run time.

Chapter 5
Discussion

This chapter discusses how the results and findings can be seen from a greater perspective, and whether they may add value to the industry and academia.

5.1 Value of increasing efficiency

The main finding from the study is increased performance in terms of less computational power required to complete the EA and CEA runs. Being able to reach the same solution in a shorter amount of time is intuitively advantageous, but will it add value to the application?

Computational power is presumably not a major bottleneck to a mining company and with the rate of technology improvement even less so. However, looking at the results from the EA, it can be seen that the same productivity can be achieved with great security and stability even after our changes to the EA were implemented. Therefore, there does not seem to be any (currently known) reasons not to implement these efficiency-enhancing alterations to the EA. For the CEA it was seen (for approach 2.4) that the number of generations could be decreased by a lot (in total 33.84 percent for problem instance 2, 3 and 6). But as this entailed a slight indication of the productivity suffering, the savings in time and computational power would probably not outweigh the (risk of) income loss.

Considering the use of this kind of scheduling method in a real-life mine, it is highly dependent on the updating of the schedules being more or less instantaneous. Being able to produce the new schedules instantly ensures that the state of the mine that the calculations are based on has not changed when the schedule is put to use. Due to this, making the algorithms more efficient is a step in the right direction.

Furthermore, considering the larger view of the impact of efficient algorithms and the role they play in the quality of the solutions, the frequency of the schedule updates needs to be considered. The current frequency, 15 minutes, is an arbitrarily chosen number that could potentially be decreased, increased or be made dependent on some other variable in the mine. The industry standard approach DISPATCH updates its suggested schedule prior to every

single truck dispatch. Given an efficient implementation of the EA, this possibility could be investigated for this approach as well. To sum up, future work should include optimisation of the schedule update frequency in one way or another.

5.2 Connection to related work

Apart from considering whether the findings motivate implementation of our changes to the EA/CEA as a standard for UWA going forward, one can consider the value these findings have in a wider context. Starting from the theoretical knowledge gathered from related work in the Literature Review (Section 2.2), the findings can be put in relation to what is said about EAs and alternative dispatching or scheduling technologies today.

As mentioned in Literature Review (Section 2.2), there are several reasons for choosing EAs to solve complex multi-objective problems over other approaches. They are suitable for handling many constraints simultaneously since they find solutions empirically, as opposed to e.g. linear programming-based scheduling technologies that are otherwise widely used. The EAs main drawback is that they require much computational power, an ever decreasing hurdle due to continuous technological development. Even so, if this hurdle can be pushed further down by other means than only awaiting technical development, that is of course of value.

Based on the same reasoning as above, using an EA to solve the problem of scheduling in mines also seems to be a sensible way to go. EAs can handle even large and complicated mine setups, where the only other feasible option is greedy heuristics as the problem setup gets complex enough. Greedy heuristics have disadvantages, as elaborated in Section 2.2, that EAs are not subject to. To sum up, the main issue limiting the scope of when EAs are the suitable option, is when they do not meet efficiency requirements. Thus, any fall forward in terms of efficiency should be embraced. Even as it is questionable if the approaches developed in this study should be put in use as they are today (e.g. considering the CEA is lacking in productivity), there should be no hesitation as to whether to have focus and resources directed here in the future. With slight fine-tuning, as described in Section 6, valuable results are anticipated for the specific problem and hopefully for the use of EAs in a greater context.

When considering how these findings can add value to EAs in a wider context, other competing approaches have to be considered. Even today, there are other ways to construct the very first generation in an EA run, in order to start off in a better place than one randomly generated. An example of this is the local search engine implemented for this purpose by Mendes *et al.* [16]. How these other potential approaches would compare to one another in terms of efficiency increase, but also difficulty level of implementation, would be an interesting research question for future work.

5.3 Connection to real life application

As mentioned in Section 1.3, the vision throughout this project has been to deliver value to the mining industry; to at some point have the findings spread to a place where they can contribute, rather than being consigned to oblivion within the academic environment of the university. The outlook of moving forward with the research to perform experiments in real

underground mines was discussed with a couple of the originators of the UWA research. The research group dedicated to this at UWA has continuously collaborated with Micromine, a mining software company specialising in mining software. The purpose of this is to stay anchored to reality and to keep an open gateway to the mining industry. Even so, real mine experiments are considered being a long way down the road. There are several hurdles that must be crossed before real-life experiments are viable. First, the mine simulator is too abstract and generic to correctly represent a real mine altogether. Also, further complexities need to be considered and incorporated into the simulator, such as machine breakdowns or staff unavailability. In addition, since mine operations are large and subject to vast monetary turnover, management are usually very risk averse. An algorithm would have to be very robust and trustworthy to ever be let in through the doors of a mining company. In conclusion, there are several barriers that are more or less difficult to climb. However, in the end it all boils down to manpower at the research project, which seems to be limited in the near future.

A large proportion of the research at UWA is financed by mining money. There are many large mining corporations in Western Australia, all with vast amounts of money and motives for investing in academic research. UWA have had relationships with such actors since more than twenty years back, which has led to the rise of a variety of projects at the university. The original ideas for such projects can stem from either UWA, a financing mining company or a third party. In this particular case the initial idea originated from Micromine, the software developer mentioned above. Micromine market themselves as a cutting-edge mining software developer, making their motivation a little ahead of the actual mine operators'. Nonetheless, Micromine (UWA's gateway to the real mines) collaborate with some progressive mining companies. Therefore, hope for an industry pull for real-life experiments and consequently implementation of EAs into the mining context prevails.

Chapter 6
Conclusion

The main findings from the experiments and the answer to the research question are as follows: The use of previous solutions from the last EA run in a single EA will result in the same peak performance compared to generating the initial schedules at random. It will increase productivity for systems with fewer trucks than in a saturated state and it will decrease the required computational power. For the approaches tried for the CEA, the peak performance will be slightly lower, as well as the productivity in an under-trucked system. The computational power will however show a massive decrease although this is still not enough to motivate a compromise with the decrease in productivity.

The findings of this study will contribute to the work at UWA going forward, and will hopefully be one component out of many to enhance the EA used for mine scheduling purposes. A recurrently mentioned drawback with EAs is the high computational power burden, why any way to increase efficiency should always be of interest. Thereby, our hope is that these findings could prove relevant even to other applications of EAs, and stand strong among contending techniques to successfully construct initial generations. Being the algorithm with the best results from previous studies, the CEA is the most interesting algorithm to take further, and there are still a lot of options to explore before the use of previous generations is discarded. The next steps would first and foremost consist of:

1. Implement an approach that only uses the previous generations for the light schedules
2. Make sure that the refitting in time does not mean inclusion of un-evaluated pieces of schedule
3. (for EA and CEA both) Combine the use of previous solutions with random generation of schedules and by doing so broadening the search space *and* preserving quality
4. Find a way to optimise the schedule updating frequency (potentially dynamically), this is a natural extension of the focus of improving the algorithm's *efficiency*.

Bibliography

[1] Chang Wook Ahn, Eungyeong Kim, Hyun-Tae Kim, Dong-Hyun Lim, and Jinung An. A hybrid multiobjective evolutionary algorithm: Striking a balance with local search. *Math. Comput. Model.*, 52(11-12):2048–2059, December 2010.

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

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

[4] Eikland K. & Notebaert P. Berkelaar, M. Lpsolve. `http://sourceforge.net/projects/lpsolve/`, June 2016 (accessed April 16, 2019).

[5] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. CRC press, Bristol, 1 edition, 2000.

[6] Francois Chollet. *Deep Learning with Python*. Manning publications co., Shelter Island, NY, 2018.

[7] W. Cox, T. French, M. Reynolds, and L. While. Applying real-time evolutionary algorithms to scheduling in underground mines. University of Western Australia, Computer Science and Software Engineering, Perth, WA, Australia, 2720, 2018.

[8] W. Cox, T. French, M. Reynolds, L. While, T. Mitrovic, Xue Bing, and Li Xiaodong. A cooperative coevolutionary algorithm for real-time underground mine scheduling. University of Western Australia, Computer Science and Software Engineering, Perth, WA, Australia, 2720, 2018.

[9] Wesley Cox, Tim French, Mark Reynolds, and Lyndon While. A genetic algorithm for truck dispatching in mining. In Christoph Benzmuller, Christine Lisetti, and Martin Theobald, editors, *GCAI 2017. 3rd Global Conference on Artificial Intelligence*, volume 50 of *EPiC Series in Computing*, pages 93–106. EasyChair, 2017.

[10] Geoff Cumming and Sue Finch. Inference by eye: Confidence intervals and how to read pictures of data. *American Psychologist*, pages 170–180, 2005.

[11] Bitanshu Das. Open pit production scheduling applying meta heuristic approach. Master's thesis, Department of Mining Engineering, National Institute of Technology, Rourkela, 2012.

[12] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, Dec 1959.

[13] S.G. Ercelebi and A. Bascetin. Optimization of shovel-truck system for surface mining. *Journal of the Southern African Institute of Mining and Metallurgy*, 109:433 – 439, 07 2009.

[14] Michel Gamache, Renaud Grimard, and Paul Cohen. A shortest-path algorithm for solving the fleet management problem in underground mines. *European Journal of Operational Research*, 166(2):497 – 506, 2005.

[15] C. W. Kim and J. M. A. Tanchoco. Conflict-free shortest-time bidirectional agv routeing. *International Journal of Production Research*, 29(12):2377 – 2391, n.d.

[16] Joao Batista Mendes, Marcos Flavio Silveira Vasconcelos D'Angelo, Nilton Alves Maia, and Rene Rodrigues Veloso. A hybrid multiobjective evolutionary algorithm for truck dispatching in open-pit-mining. *IEEE Latin America Transactions*, 14(3):1329, 2016.

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

[18] Christie Myburgh and Kalyanmoy Deb. Evolutionary algorithms in large-scale open pit mine scheduling. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 1155–1162, New York, NY, USA, 2010. ACM.

[19] Y. Osada, Lyndon While, L. Barone, and Z. Michalewicz. Multi-mine planning using a multi-objective evolutionary algorithm. In *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 2902 – 2909, United States, 2013. IEEE, Institute of Electrical and Electronics Engineers.

[20] Irem Ozkarahan, Pinar Mizrak Ozfirat, and M. Kemal Ozfirat. A mathematical modeling approach for the scheduling problem of load-haul-dump vehicles in underground mines. *Proceedings for the Northeast Region Decision Sciences Institute (NEDSI)*, pages 1136 – 1143, 2011.

[21] Mitchell A. Potter and Kenneth A. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, PPSN III, pages 249–257, London, UK, UK, 1994. Springer-Verlag.

[22] J.-Y. Potvin. State-of-the art review:evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing*, 21(4):518–548, 2009.

[23] S. Rupprecht. Mine development–access to deposit. In *5th International Platinum Conference*, pages 101–121, 2012.

[24] Modular Mining Systems. Dispatch underground mine management system. `https://www.modularmining.com/product/dispatch-underground/`, 2019 (accessed April 10, 2019).

[25] Modular Mining Systems. Our solutions, load and haul. `https://www.modularmining.com/our-solutions/load-and-haul/`, 2019 (accessed April 10, 2019).

[26] Ramani R. V. Tan, S. Multi-mine planning using a multi-objective evolutionary algorithm. In *Evaluation of computer truck dispatching criteria*, pages 192 – 215, United States, 1992. InProceedings of the SME/AIME annual meeting and exhibition.

[27] Otuonye F.O. Temeng, V.A. and J.O. Frendewey. A nonpreemptive goal programming approach to truck dispatching in open pit mines. *Mineral Resources Engineering*, (7):59 – 67, 1998.

[28] Lyndon While. Cits4404 artificial intelligence and adaptive systems, ci technologies. University Lecture, 2019. Department of Computer Science, University of Western Australia.

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

[30] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, pages 292–301, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

Appendices

Appendix A
Complete results

A.1 Evolutionary algorithm

A.1.1 Summary results

Productivity

Table A.1: EA. Problem instance 1. Productivity (number of truck-loads), standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
12	134.20	1.30	134.44	1.24	0.18%	134.76	1.07	0.42%
13	140.48	1.02	140.92	0.80	0.31%	139.36	0.93	-0.80%
14	145.24	0.65	145.40	0.69	0.11%	145.32	0.88	0.06%
15	150.48	0.85	151.00	0.69	0.35%	150.96	0.82	0.32%
16	153.88	0.77	153.84	0.46	-0.03%	153.86	0.64	-0.01%
17	154.84	0.37	154.64	0.56	-0.13%	154.84	0.46	0.00%
18	154.84	0.37	154.84	0.37	0.00%	154.96	0.34	0.08%
19	155.00	0.49	155.00	0.28	0.00%	155.00	0.28	0.00%
20	155.20	0.49	155.40	0.49	0.13%	155.56	0.50	0.23%
21	155.52	0.50	155.36	0.56	-0.10%	155.60	0.49	0.05%
Total	1499.68		1500.84			1500.22		
Average		0.68		0.61			0.64	
Change					0.08%			0.04%

Table A.2: EA. Problem instance 2. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
14	137.88	0.99	137.44	0.98	-0.32%	138.40	0.75	0.38%
15	143.36	0.74	144.84	0.73	1.03%	144.64	0.84	0.89%
16	149.56	0.70	149.56	0.80	0.00%	149.44	0.75	-0.08%
17	152.76	0.65	153.08	0.63	0.21%	152.72	0.66	-0.03%
18	153.76	0.43	153.84	0.37	0.05%	153.92	0.39	0.10%
19	154.76	0.51	154.04	0.20	-0.47%	154.84	0.46	0.05%
20	154.92	0.27	155.04	0.45	0.08%	155.04	0.34	0.08%
21	155.04	0.20	155.04	0.34	0.00%	155.04	0.20	0.00%
22	155.00	0.57	154.92	0.39	-0.05%	155.04	0.45	0.03%
23	154.92	0.27	154.88	0.52	-0.03%	155.00	0.00	0.05%
Total	1511.96		1512.68			1514.08		
Average		0.53		0.54			0.48	
Change					0.05%			0.14%

Table A.3: EA. Problem instance 3. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
13	131.76	0.65	133.60	1.10	1.40%	134.16	0.97	1.82%
14	138.48	0.75	140.08	0.84	1.16%	138.52	0.90	0.03%
15	145.84	0.88	146.92	0.89	0.74%	146.44	1.13	0.41%
16	151.56	0.75	151.76	0.59	0.13%	151.84	0.67	0.18%
17	153.32	0.73	153.80	0.49	0.31%	153.84	0.78	0.34%
18	153.84	0.54	154.12	0.65	0.18%	154.12	0.52	0.18%
19	154.20	0.49	154.36	0.56	0.10%	154.12	0.43	-0.05%
20	154.76	0.65	155.00	0.57	0.16%	154.88	0.43	0.08%
21	155.20	0.57	155.08	0.39	-0.08%	155.16	0.46	-0.03%
22	155.20	0.57	154.96	0.45	-0.15%	155.20	0.49	0.00%
Total	1494.16		1499.68			1498.28		
Average		0.66		0.65			0.68	
Change					0.37%			0.28%

Table A.4: EA. Problem instance 4. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
11	131.48	0.94	132.00	1.52	0.40%	133.04	1.78	1.19%
12	135.96	0.66	138.40	0.85	1.79%	138.24	1.07	1.68%
13	144.40	1.02	145.16	0.61	0.53%	145.16	0.92	0.53%
14	150.04	0.82	150.80	0.63	0.51%	150.72	0.83	0.45%
15	153.88	0.91	153.84	0.67	-0.03%	154.16	0.73	0.18%
16	155.04	0.45	155.28	0.66	0.15%	155.36	0.56	0.21%
17	155.68	0.47	155.80	0.40	0.08%	155.64	0.48	-0.03%
18	155.68	0.47	155.60	0.49	-0.05%	155.64	0.48	-0.03%
19	155.72	0.45	155.80	0.40	0.05%	155.72	0.45	0.00%
20	155.72	0.45	155.72	0.66	0.00%	155.44	0.50	-0.18%
Total	1493.60		1498.4			1499.12		
Average		0.66		0.69			0.78	
Change					0.32%			0.37%

Table A.5: EA. Problem instance 5. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
12	130.56	0.92	130.48	0.85	-0.06%	130.08	1.13	-0.37%
13	137.48	0.70	138.00	0.89	0.38%	137.76	1.07	0.20%
14	143.92	0.89	144.32	0.97	0.28%	144.68	0.73	0.53%
15	150.44	0.98	150.12	1.11	-0.21%	150.16	0.73	-0.19%
16	153.48	0.57	153.56	0.50	0.05%	153.44	0.64	-0.03%
17	155.16	0.46	155.16	0.46	0.00%	155.24	0.43	0.05%
18	155.12	0.43	155.36	0.56	0.15%	155.40	0.49	0.18%
19	155.32	0.55	155.44	0.50	0.08%	155.48	0.50	0.10%
20	155.36	0.48	155.56	0.50	0.13%	155.44	0.50	0.05%
21	155.52	0.50	155.48	0.50	-0.03%	155.56	0.50	0.03%
Total	1492.36		1493.48			1493.24		
Average		0.65		0.68			0.67	
Change					0.08%			0.06%

Table A.6: EA. Problem instance 6. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
13	132.12	1.03	134.32	1.16	1.67%	133.64	0.79	1.15%
14	138.72	0.83	140.52	0.98	1.30%	140.52	0.98	1.30%
15	145.48	0.85	146.96	0.72	1.02%	145.52	1.10	0.03%
16	151.52	0.64	151.96	0.96	0.29%	152.00	0.85	0.32%
17	153.68	0.55	153.84	0.61	0.10%	153.60	0.49	-0.05%
18	154.64	0.69	154.40	0.57	-0.16%	154.72	0.53	0.05%
19	155.00	0.00	155.08	0.39	0.05%	154.92	0.39	-0.05%
20	155.12	0.43	155.00	0.40	-0.08%	154.84	0.46	-0.18%
21	154.96	0.34	155.12	0.43	0.10%	154.92	0.48	-0.03%
22	155.16	0.37	155.16	0.54	0.00%	154.84	0.46	-0.21%
Total	1496.40		1502.36			1499.52		
Average		0.57		0.68			0.65	
Change					0.40%			0.21%

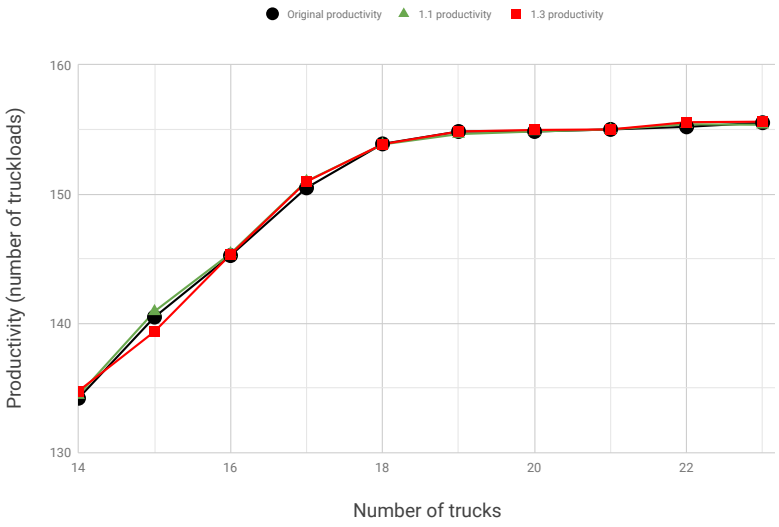


Figure A.1: EA. Problem instance 1. Average mine productivity for the approaches respectively by the number of trucks in use.

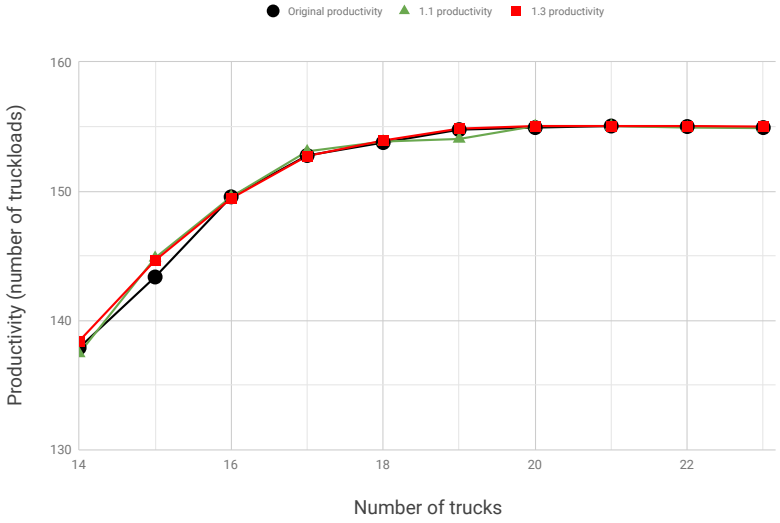


Figure A.2: EA. Problem instance 2. Average mine productivity for the approaches respectively by the number of trucks in use.

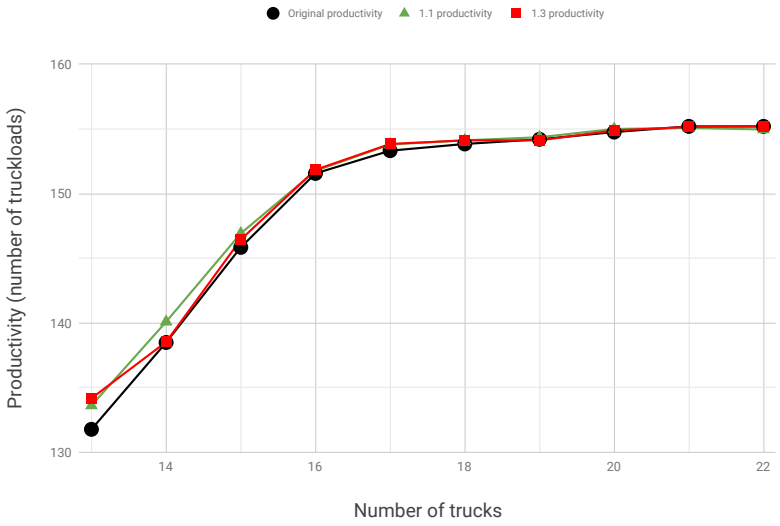


Figure A.3: EA. Problem instance 3. Average mine productivity for the approaches respectively by the number of trucks in use.

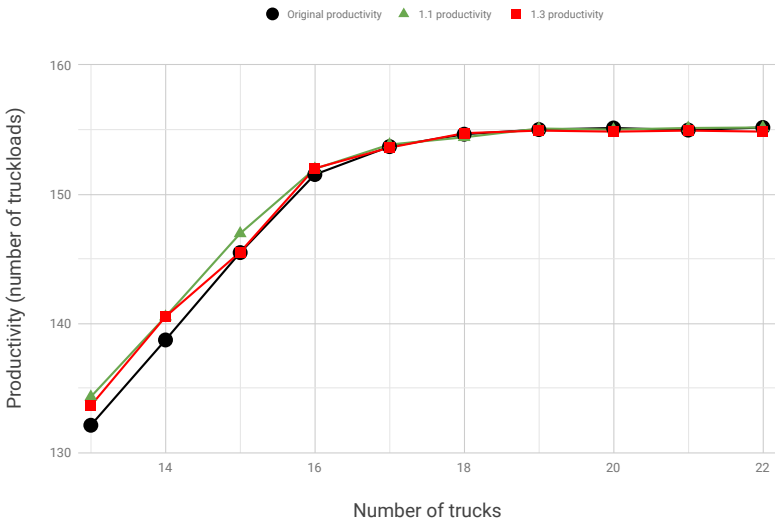


Figure A.4: EA. Problem instance 4. Average mine productivity for the approaches respectively by the number of trucks in use.

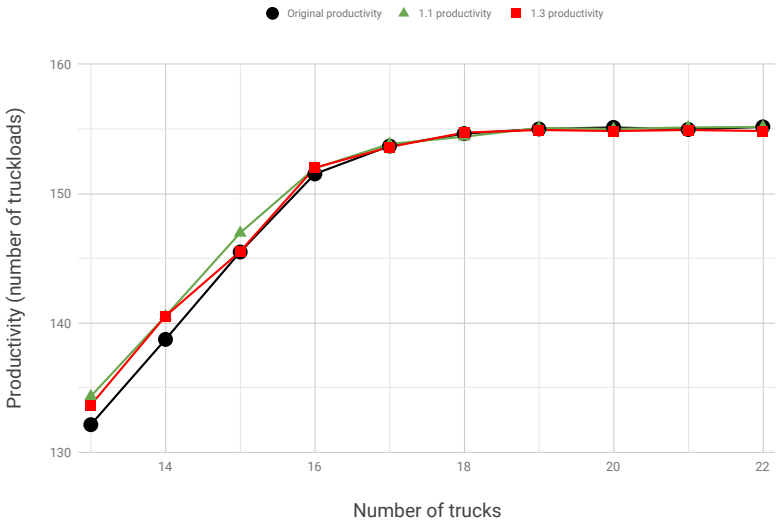


Figure A.5: EA. Problem instance 5. Average mine productivity for the approaches respectively by the number of trucks in use.

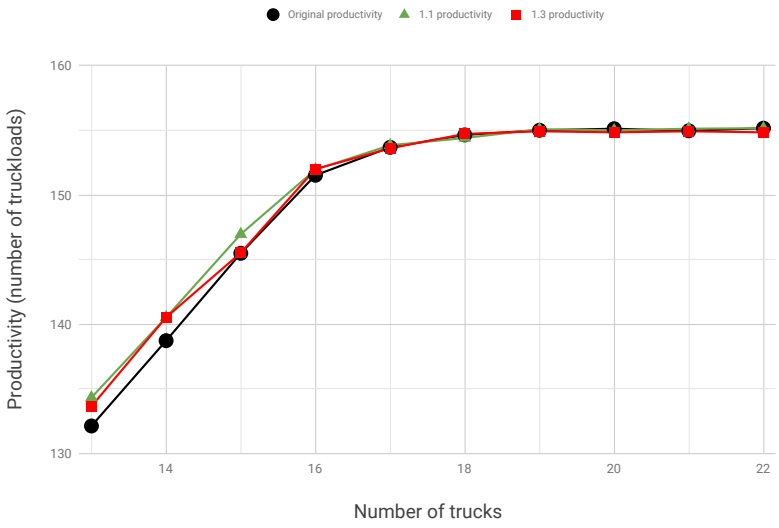


Figure A.6: EA. Problem instance 6. Average mine productivity for the approaches respectively by the number of trucks in use.

Number of generations

Table A.7: EA. Problem instance 1. Number of generations, standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
12	213.08	59.30	191.32	51.40	-10.21%	194.78	51.24	-8.59%
13	227.13	62.23	203.91	56.15	-10.22%	210.88	62.92	-7.15%
14	236.04	65.45	218.79	65.14	-7.31%	212.15	62.01	-10.12%
15	258.07	80.04	225.21	74.20	-12.73%	226.92	67.58	-12.07%
16	226.52	89.25	213.24	69.53	-5.86%	203.25	75.93	-10.27%
17	163.14	97.87	145.62	99.12	-10.74%	154.34	99.27	-5.40%
18	122.66	102.18	115.36	107.00	-5.95%	119.27	102.96	-2.76%
19	113.36	104.18	106.07	104.39	-6.43%	104.37	104.29	-7.92%
20	90.42	108.75	128.83	102.21	42.48%	119.63	103.08	32.31%
21	135.02	101.49	113.76	109.79	-15.74%	120.92	100.25	-10.44%
Total	1785.42		1662.11			1666.50		
Average		87.07		83.89			82.95	
Change					-6.91%			-6.66%

Table A.8: EA. Problem instance 2. Number of generations, standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
14	218.74	54.89	205.36	55.28	-6.12%	211.84	60.01	-3.15%
15	236.50	64.48	217.59	64.80	-8.00%	225.00	66.38	-4.86%
16	252.60	74.27	229.04	72.39	-9.33%	227.26	72.00	-10.03%
17	252.85	80.32	225.34	77.33	-10.88%	219.33	68.77	-13.26%
18	198.94	98.84	164.42	101.69	-17.35%	157.00	102.39	-21.08%
19	154.46	101.01	132.26	103.14	-14.37%	130.63	108.53	-15.43%
20	137.96	111.59	121.46	100.29	-11.96%	125.48	105.89	-9.05%
21	138.46	104.61	123.00	102.10	-11.17%	102.41	98.05	-26.04%
22	102.85	100.45	92.52	101.62	-10.04%	98.78	104.00	-3.95%
23	122.05	106.45	101.96	99.74	-16.46%	101.21	106.82	-17.07%
Total	1815.40		1612.94			1598.94		
Average		89.69		87.84			89.28	
Change					-11.15%			-11.92%

Table A.9: EA. Problem instance 3. Number of generations, standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
13	210.07	53.12	202.01	57.94	-3.84%	203.25	58.70	-3.24%
14	224.99	55.96	221.83	68.45	-1.41%	207.09	61.39	-7.96%
15	257.30	74.60	221.57	63.84	-13.89%	231.22	74.02	-10.14%
16	262.30	78.08	222.40	71.74	-15.21%	222.95	70.55	-15.00%
17	235.89	85.57	181.50	89.64	-23.06%	166.43	88.19	-29.44%
18	163.61	105.35	98.41	99.63	-39.85%	131.42	102.41	-19.67%
19	126.57	105.63	96.27	95.46	-23.94%	92.74	98.91	-26.73%
20	162.08	95.11	148.80	94.50	-8.19%	155.74	95.21	-3.91%
21	132.75	98.88	134.99	97.98	1.68%	132.33	98.47	-0.32%
22	145.83	95.80	136.57	95.91	-6.35%	126.91	99.83	-12.97%
Total	1921.39		1664.35			1670.09		
Average		84.81		83.51			84.77	
Change					-13.38%			-13.08%

Table A.10: EA. Problem instance 4. Number of generations, standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
11	213.32	52.81	185.92	50.08	-12.85%	185.67	47.39	-12.96%
12	218.47	59.67	202.49	59.22	-7.32%	205.90	59.09	-5.76%
13	246.16	68.50	214.54	62.89	-12.85%	217.38	62.62	-11.69%
14	256.26	78.90	218.65	70.01	-14.68%	227.00	69.67	-11.42%
15	250.67	77.65	213.22	64.66	-14.94%	219.45	70.76	-12.45%
16	180.30	97.80	166.46	94.50	-7.68%	148.87	99.85	-17.44%
17	146.96	111.59	114.86	102.38	-21.84%	114.79	102.87	-21.89%
18	149.92	113.36	94.02	101.68	-37.29%	106.68	104.74	-28.84%
19	112.47	116.14	101.61	105.73	-9.66%	123.05	104.77	9.41%
20	99.45	106.40	127.24	100.00	27.94%	107.90	102.78	8.50%
Total	1873.98		1639.01			1656.70		
Average		88.28		81.11			82.45	
Change					-12.54%			-11.06%

Table A.11: EA. Problem instance 5. Number of generations, standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
12	197.82	45.87	191.11	53.89	-3.39%	185.76	46.02	-6.10%
13	215.47	55.16	201.48	57.15	-6.49%	202.97	55.63	-5.80%
14	243.77	65.06	203.71	60.34	-16.43%	208.77	59.98	-14.36%
15	262.32	74.29	226.52	68.63	-13.65%	227.51	73.29	-13.27%
16	254.74	75.14	217.78	73.54	-14.51%	207.60	74.43	-18.51%
17	165.44	107.06	129.23	100.02	-21.89%	144.97	102.44	-12.38%
18	111.83	104.03	88.53	97.16	-20.84%	115.54	101.17	3.32%
19	122.64	102.61	105.99	100.75	-13.58%	114.75	99.78	-6.44%
20	99.42	102.04	89.06	99.52	-10.42%	104.23	103.10	4.84%
21	112.29	104.04	97.50	101.65	-13.17%	92.76	100.90	-17.39%
Total	1785.75		1550.91			1604.86		
Average		83.53		81.27			81.67	
Change					-13.15%			-10.13%

Table A.12: EA. Problem instance 6. Number of generations, standard deviation and difference compared to original.

	Original		Approach 1.1			Approach 1.3		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
13	203.65	48.14	189.02	50.54	-7.19%	185.07	47.16	-9.12%
14	224.02	60.89	200.92	59.81	-10.31%	198.81	56.53	-11.25%
15	246.97	75.00	207.89	58.71	-15.82%	222.44	65.98	-9.93%
16	252.00	72.10	216.81	64.25	-13.96%	216.38	61.96	-14.13%
17	238.29	69.83	213.15	69.87	-10.55%	206.34	72.12	-13.41%
18	197.95	99.90	192.63	78.23	-2.69%	163.74	96.65	-17.28%
19	174.55	102.93	145.63	101.59	-16.57%	144.48	105.36	-17.23%
20	161.79	103.51	120.05	103.38	-25.80%	113.13	106.85	-30.08%
21	152.23	101.83	128.76	96.55	-15.42%	127.07	98.64	-16.53%
22	128.47	102.40	117.20	101.40	-8.77%	135.29	98.75	5.31%
Total	1979.92		1732.06			1712.75		
Average		83.65		78.43			81.00	
Change					-12.52%			-13.49%

Figure A.7: EA. Problem instance 1. Average number of generations for the approaches respectively by the number of trucks in use.

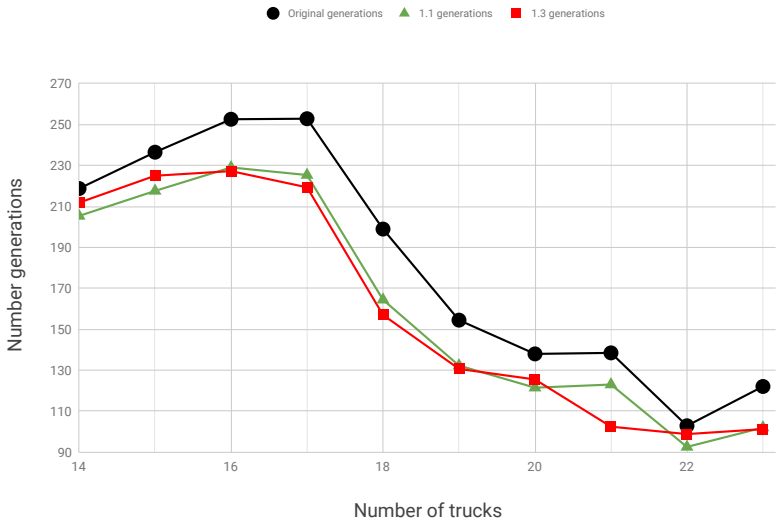


Figure A.8: EA. Problem instance 2. Average number of generations for the approaches respectively by the number of trucks in use.

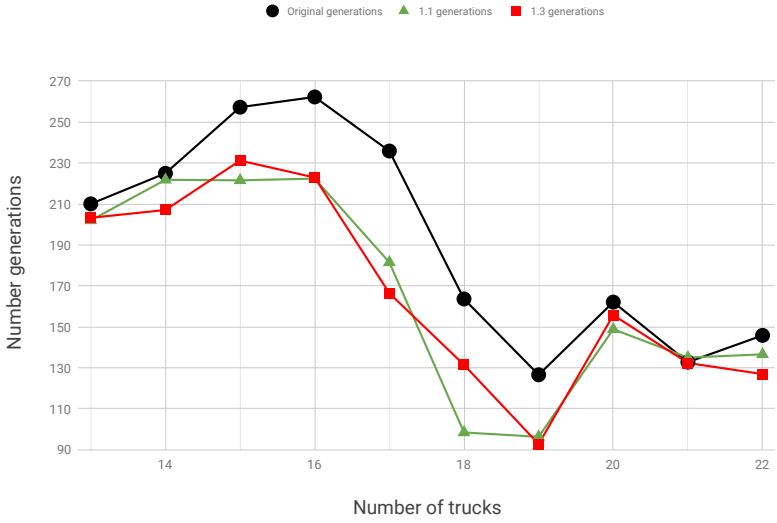


Figure A.9: EA. Problem instance 3. Average number of generations for the approaches respectively by the number of trucks in use.

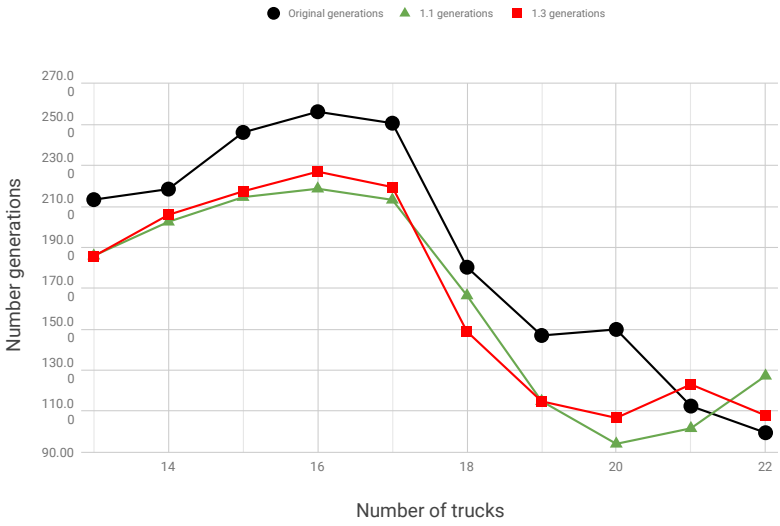


Figure A.10: EA. Problem instance 4. Average number of generations for the approaches respectively by the number of trucks in use.

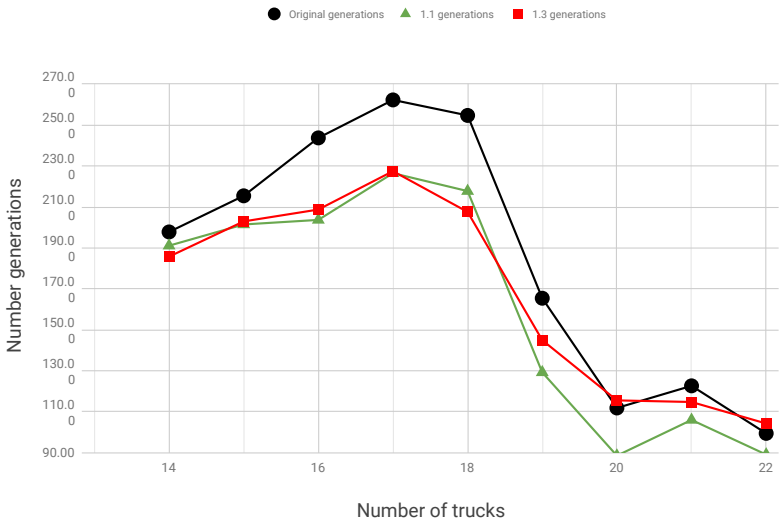


Figure A.11: EA. Problem instance 5. Average number of generations for the approaches respectively by the number of trucks in use.

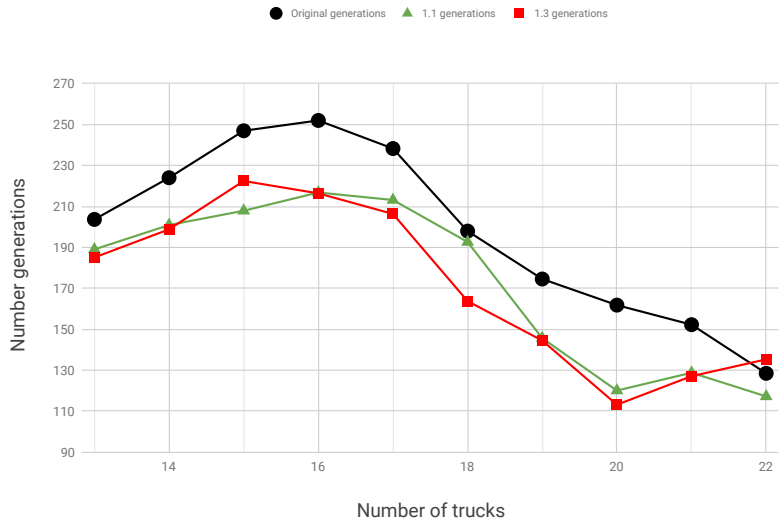


Figure A.12: EA. Problem instance 6. Average number of generations for the approaches respectively by the number of trucks in use.

A.2 Co-evolutionary algorithm

A.2.1 Summary results

Productivity

Table A.13: CEA. Problem instance 2. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	Prod	St dev	Prod	St dev	Diff	Prod	St dev	Diff
14	140.80	0.5800	140.76	0.66	-0.03%	139.48	1.2600	-0.94%
15	147.08	0.57	147.20	0.71	0.08%	145.04	4.6000	-1.39%
16	151.52	0.77	151.48	0.59	-0.03%	150.96	0.9300	-0.37%
17	154.00	0.29	153.96	0.54	-0.03%	153.48	0.5900	-0.34%
18	154.52	0.51	154.64	0.49	0.08%	154.36	0.6400	-0.10%
19	154.84	0.37	155.00	0.41	0.10%	154.56	0.5800	-0.18%
20	155.00	0.41	155.16	0.47	0.10%	154.64	0.5700	-0.23%
21	155.08	0.4000	155.04	0.35	-0.03%	154.64	0.6400	-0.28%
22	155.20	0.41	155.04	0.20	-0.10%	154.80	0.4100	-0.26%
23	155.20	0.5	155.12	0.33	-0.05%	154.48	0.8200	-0.46%
Total	1523.24		1523.40			1516.44		
Average		0.48		0.48			1.10	
Change					0.01%			-0.45%

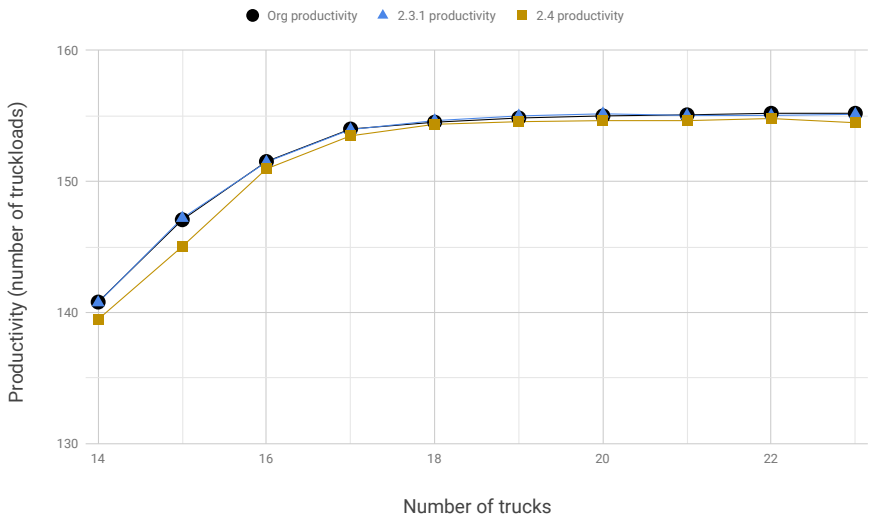


Figure A.13: CEA. Problem instance 2. Average mine productivity for the approaches respectively by the number of trucks in use.

Table A.14: CEA. Problem instance 3. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
13	136.28	0.61	136.40	0.76	0.09%	133.32	0.9500	-2.17%
14	142.76	0.6	142.88	0.83	0.08%	141.20	0.7600	-1.09%
15	148.52	0.7100	148.36	0.70	-0.11%	147.48	0.8700	-0.70%
16	152.88	0.67	152.72	0.68	-0.10%	152.32	0.7500	-0.37%
17	154.24	0.52	154.32	0.48	0.05%	154.08	0.4900	-0.10%
18	154.72	0.54	154.80	0.50	0.05%	154.64	0.4900	-0.05%
19	155.08	0.49	154.96	0.36	-0.08%	154.76	0.4400	-0.21%
20	155.24	0.44	155.28	0.46	0.03%	154.72	0.6800	-0.33%
21	155.12	0.33	155.04	0.35	-0.05%	154.96	0.7900	-0.10%
22	155.36	0.49	155.08	0.49	-0.18%	154.60	0.9100	-0.49%
Total	1510.20		1509.83			1502.08		
Average		0.54		0.56			0.71	
Change					-0.02%			-0.54%

Table A.15: CEA. Problem instance 6. Productivity (number of truckloads), standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	**Prod**	**St dev**	**Prod**	**St dev**	**Diff**	**Prod**	**St dev**	**Diff**
13	136.12	0.6	136.20	0.71	0.06%	133.88	2.19	-1.65%
14	142.44	0.51	142.44	0.58	0.00%	141.24	1.42	-0.84%
15	147.64	0.57	147.72	0.61	0.05%	146.96	0.73	-0.46%
16	152.52	0.59	152.52	0.59	0.00%	151.92	1.35	-0.39%
17	154.16	0.37	154.04	0.61	-0.08%	153.56	0.82	-0.39%
18	154.80	0.41	154.72	0.54	-0.05%	153.68	0.9	-0.72%
19	155.00	0.41	154.88	0.44	-0.08%	154.52	0.65	-0.31%
20	155.08	0.28	155.16	0.47	0.05%	154.36	0.81	-0.46%
21	155.12	0.33	155.20	0.50	0.05%	154.52	0.92	-0.39%
22	155.24	0.44	155.20	0.50	-0.03%	154.36	0.86	-0.57%
Total	1508.12		1508.08			1499.00		
Average		0.45		0.56			1.07	
Change					0.00%			-0.60%

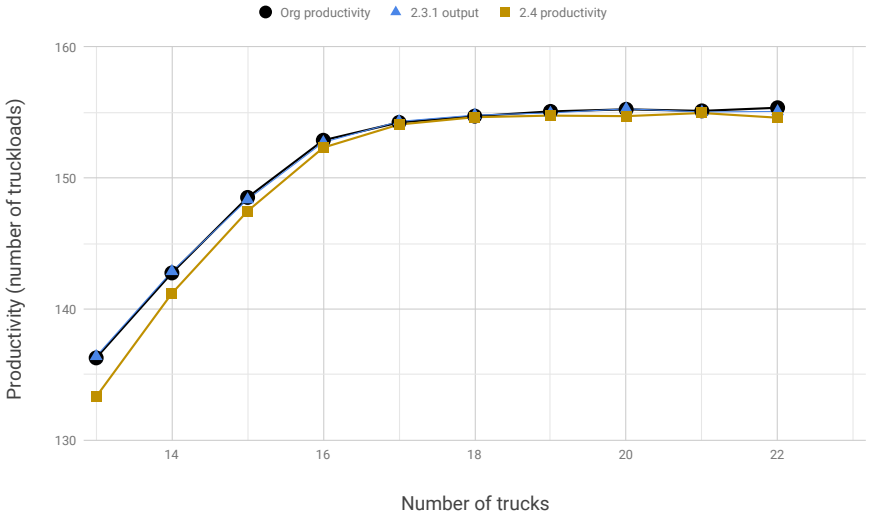


Figure A.14: CEA. Problem instance 3. Average mine productivity for the approaches respectively by the number of trucks in use.

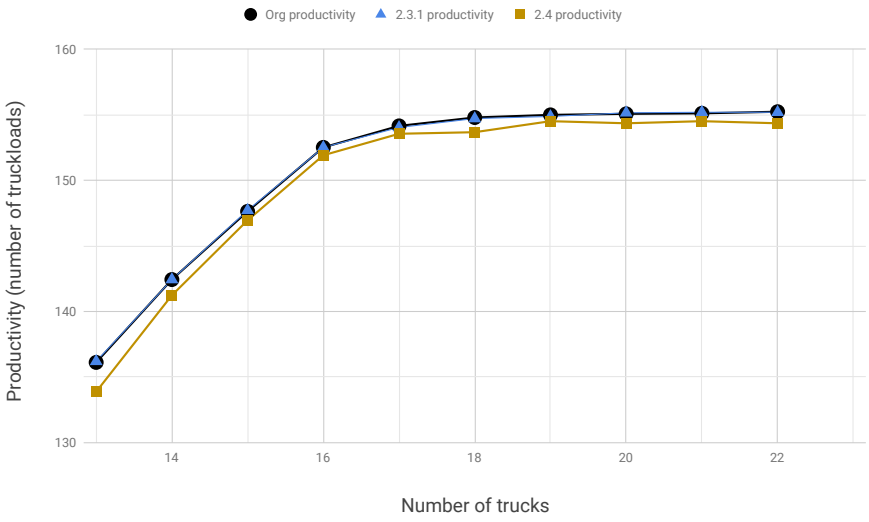


Figure A.15: CEA. Problem instance 6. Average mine productivity for the approaches respectively by the number of trucks in use.

Number of generations

Table A.16: CEA. Problem instance 2. Number of generations, standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
14	385.15	91.74	381.35	101.05	-0.99%	268.64	87.71	-30.25%
15	403.09	105.53	403.37	103.26	0.07%	282.42	93.81	-29.94%
16	417.94	115.92	418.49	116.33	0.13%	290.63	92.45	-30.46%
17	395.42	115.07	395.93	131.28	0.13%	257.18	110.41	-34.96%
18	337.53	125.75	327.92	126.91	-2.85%	198.71	126.05	-41.13%
19	275.04	142.80	264.83	147.37	-3.71%	153.50	139.26	-44.19%
20	222.41	144.09	224.92	135.58	1.13%	129.03	137.82	-41.99%
21	231.81	142.21	227.32	141.27	-1.94%	100.74	129.60	-56.54%
22	232.70	136.54	222.06	146.15	-4.57%	114.56	130.32	-50.77%
23	237.70	137.41	223.34	141.08	-6.04%	127.30	137.61	-46.45%
Total	3138.79		3089.53			1922.71		
Average		125.71		129.028			118.504	
Change					-1.57%			-38.74%

Table A.17: CEA. Problem instance 3. Number of generations, standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
13	384.44	94.98	386.05	101.15	0.42%	262.22	80.50	-31.79%
14	420.62	104.43	431.39	100.72	2.56%	288.06	95.50	-31.52%
15	448.34	113.48	455.88	117.40	1.68%	290.74	93.27	-35.15%
16	426.99	123.31	425.76	116.20	-0.29%	299.90	100.90	-29.76%
17	341.62	136.73	339.20	140.62	-0.71%	255.01	120.70	-25.35%
18	308.34	147.27	345.17	126.11	11.94%	200.98	127.89	-34.82%
19	311.58	132.14	275.93	144.39	-11.44%	180.13	132.78	-42.19%
20	223.30	140.75	228.17	145.55	2.18%	158.24	140.93	-29.14%
21	224.46	134.87	231.44	139.16	3.11%	157.08	132.68	-30.02%
22	235.05	145.60	246.62	131.91	4.92%	112.28	134.30	-52.23%
Total	3324.74		3365.61			2204.64		
Average		127.36		126.32			115.95	
Change					1.23%			-33.69%

Table A.18: CEA. Problem instance 6. Number of generations, standard deviation and difference compared to original.

	Original		Approach 2.3.1			Approach 2.4		
Trucks	**Gen**	**St dev**	**Gen**	**St dev**	**Diff**	**Gen**	**St dev**	**Diff**
13	344.69	93.54	344.65	89.37	-0.01%	237.84	81.15	-31.00%
14	366.92	97.08	373.94	101.87	1.91%	270	94.18	-26.41%
15	404.27	106.65	408.22	114.24	0.98%	282.87	98.03	-30.03%
16	396.02	125.73	400.00	130.46	1.00%	295.16	104.6	-25.47%
17	310.35	134.9	315.89	145.64	1.79%	254.61	107.47	-17.96%
18	278.04	153.76	290.36	144.86	4.43%	226.75	115.1	-18.45%
19	236.63	149.69	246.83	142.68	4.31%	202.22	129.04	-14.54%
20	211.73	145.63	239.51	145.09	13.12%	143.36	134.08	-32.29%
21	248.79	139.53	221.35	143.14	-11.03%	141.53	138.24	-43.11%
22	246.34	140.02	236.20	146.76	-4.12%	108.8	128.04	-55.83%
Total	3043.78		3076.95			2163.14		
Average		128.65		130.41			112.99	
Change					1.09%			-28.93%

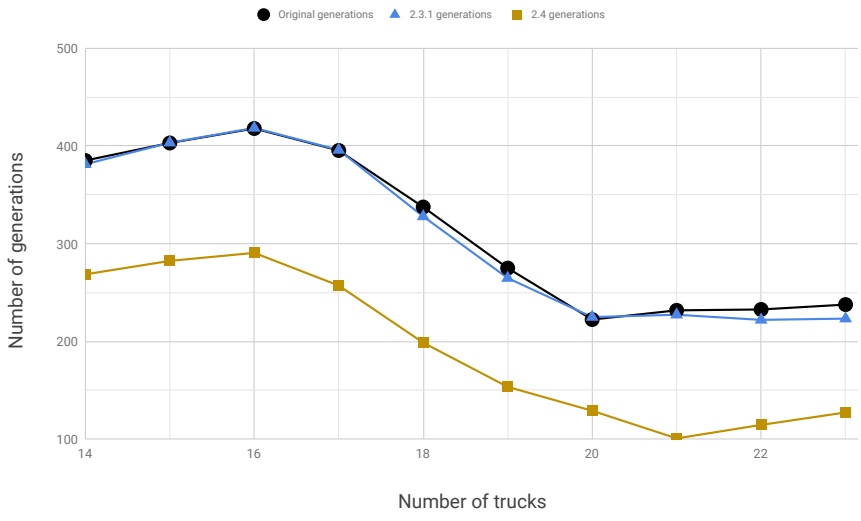


Figure A.16: CEA. Problem instance 2. Average number of generations for the approaches respectively by the number of trucks in use.

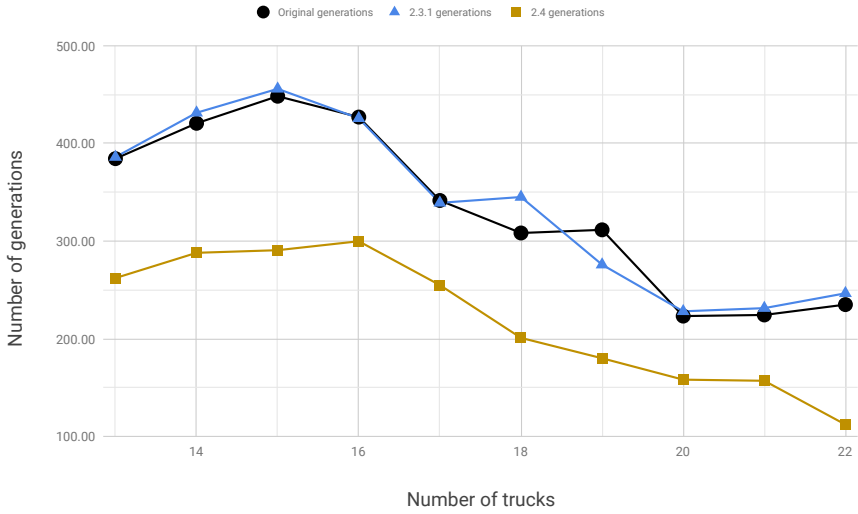


Figure A.17: CEA. Problem instance 3. Average number of generations for the approaches respectively by the number of trucks in use.

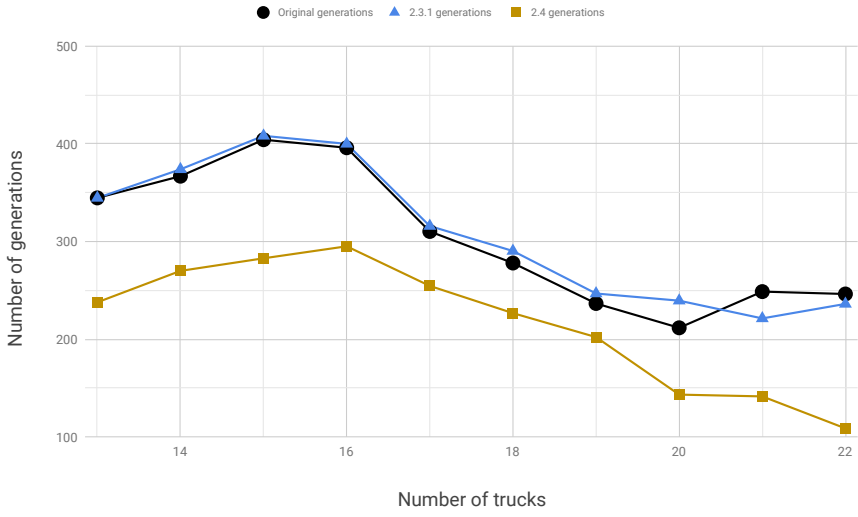


Figure A.18: CEA. Problem instance 6. Average number of generations for the approaches respectively by the number of trucks in use.

A.2.2 Result analysis

Stability analysis of output

Performed for CEA only, since the CEA was suspected to be prone to produce outliers.

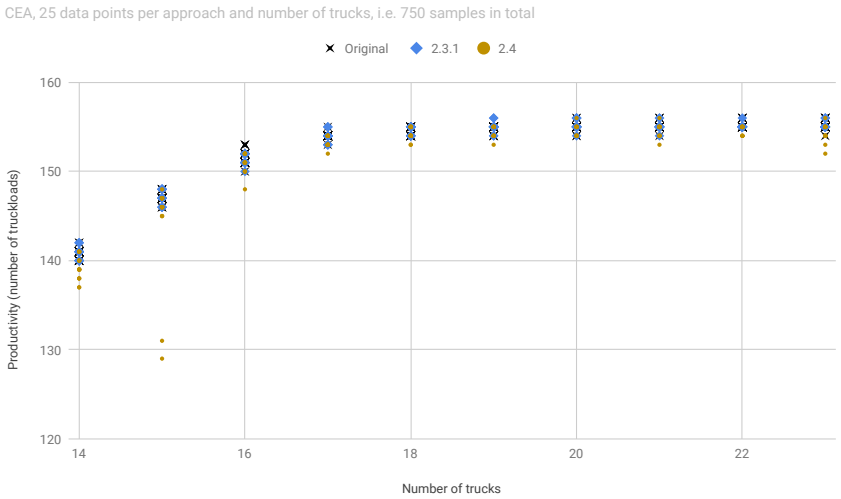


Figure A.19: CEA. Problem instance 2. Number of truckloads by number of trucks in use and approach (25 samples per number of trucks). Highlights the two outliers produced by approach 2.4 for 15 trucks.

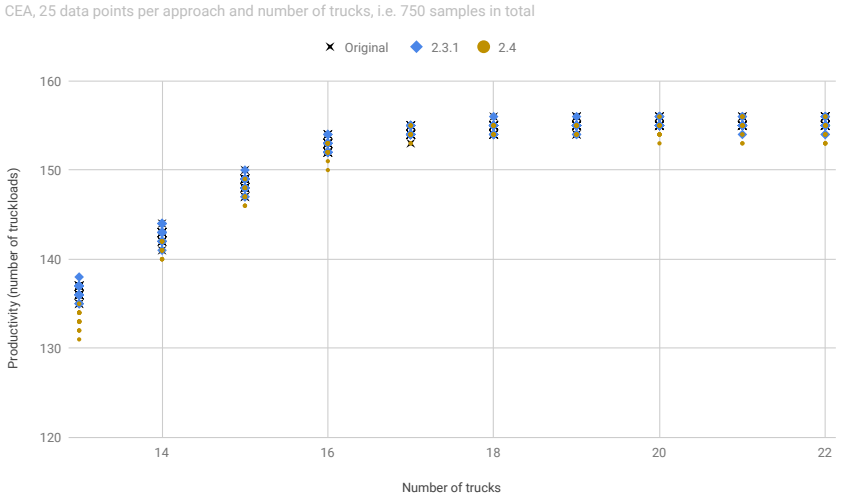


Figure A.20: CEA. Problem instance 3. Number of truckloads by number of trucks in use and approach (25 samples per number of trucks). Displays a lower productivity for 2.4.

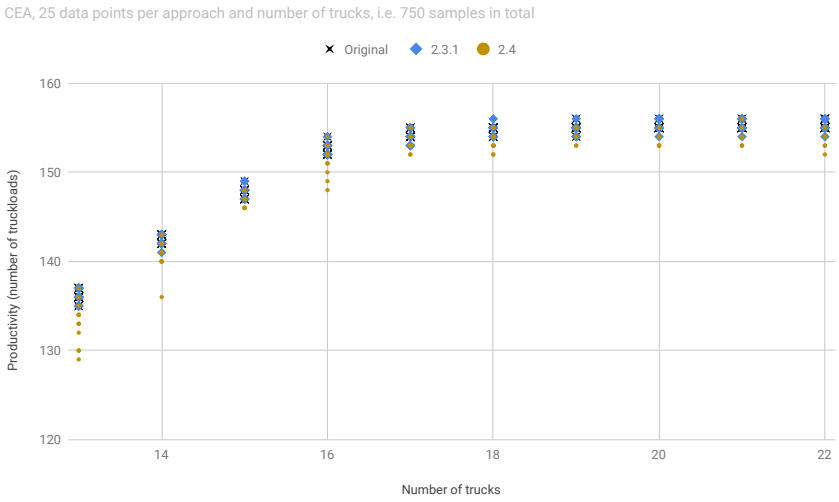


Figure A.21: CEA. Problem instance 6. Number of truckloads by number of trucks in use and approach (25 samples per number of trucks). Displays both a lower productivity and unstability for 2.4

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2019-05-31

EXAMENSARBETE Improving Real-time Scheduling by Evolutionary Algorithms
STUDENTER Hedda Malmström, Sofia Tenerz
HANDLEDARE Elin Anna Topp (LTH)
EXAMINATOR Jacek Malec (LTH)

Kan användning av gamla scheman underlätta sökningen efter nya?

POPULÄRVETENSKAPLIG SAMMANFATTNING **Hedda Malmström, Sofia Tenerz**

Att schemalägga lastbilsrutter i gruvor är ett avancerat problem vars svårighetsgrad växer snabbt i takt med antalet lastbilar och komplexiteten i vägsystemen under jordytan. Tidigare forskning har använt sig av artificiell intelligens för att producera scheman för lastbilarna. Att räkna fram helt nya körplaner tar tid, men genom att använda information från redan använda scheman kan man ofta få likvärdiga resultat, snabbare.

Det finns många metoder som kan användas för schemaläggning, både i det generella fallet och när man pratar om gruvor. Klassiska metoder är antingen *deterministiska*, dvs. man räknar ut det verkligt optimala schemat, eller *heuristiska*, dvs. man använder en enkel prioriteringsregel (t.ex. lastbilar får åka i den ordning de kom). För ett väldigt komplext problem kan en deterministisk metod behöva tio- eller hundratals år för att komma fram till ett resultat om det någonsin ens är möjligt. Genom artificiell intelligens kan man arbeta *empiriskt* och testa sig fram och på så sätt ofta nå bättre lösningar.

En algoritm som tidigare använts för att producera lastbilsscheman i gruvor kallas "evolutionär algoritm" och efterliknar sättet som naturen förädlar arter; de starkaste individerna överlever och får föröka sig. Detta sätt att söka efter lösningar på ett problem har visat sig vara bra, resultatet är oftast bättre än vad man kan åstadkomma med heuristiska och deterministiska metoder.

En av nackdelarna med att använda evolutionära algoritmer är att det krävs förhållandevis mycket beräkningskraft och tid för att komma fram till en lösning. Jämför med hur lång tid det tagit för mänskligheten att komma dit vi är idag!

För att uppnå bästa möjliga resultat måste man dessutom uppdatera körplanerna ofta. Genom att ta det gamla schemat som utgångspunkt när man söker efter nästa kan man korta ner tiden det tar att hitta en ny plan för hur lastbilarna ska köra. Det höga värdet på malmen som bryts gör det emellertid inte värt att göra avkall på gruvans produktivitet för att spara in datakraft.

När man använder sig av evolutionära algoritmer är utseendet hos den första generationen av individer i en population avgörande för hur den slutgiltiga lösningen ser ut. Är alla "föräldrar" korta finns det dåliga förutsättningar för att deras "avkommor" ska bli långa, trots att det kanske hade varit positivt om maten hänger på höga träd! Det är därför viktigt att tänka efter när man sätter förutsättningarna för algoritmens sökning. Denna studie har visat att man kan effektivisera sökandet efter lösningar genom att utgå ifrån gamla scheman, och därmed begränsa mängden nya egenskaper i den första generationen. Effektivitetsökningen kan dessutom ofta uppnås utan att kompromissa på produktiviteten!