The Application of a hybrid Evolutionary Algorithm to the Train Unit Shunting Problem

van Bavel, Niek C.L.

Award date:
2022

MASTER THESIS

OPERATIONS MANAGEMENT & LOGISTICS

The Application of a hybrid Evolutionary Algorithm to the Train Unit Shunting Problem.

Author:
N.C.L. van Bavel

Supervisors:
L. Bliek, TU/e
R. van den Broek, NS
I. Grau Garcia, TU/e

July 5, 2022

Abstract

The Train Unit Shunting Problem (TUSP) is the problem of finding feasible solutions for the matching, parking, routing, and servicing of train units at a shunting yard. The objective of this research is to effectively configure an evolutionary algorithm (EA) to generate feasible solutions for TUSP. Previous work has failed to optimize a meta-heuristic able to tackle the complete sub-problems of TUSP including the option to split and combine train units. This research introduces a recombination technique to connect the most promising parts of two parents into the new offspring solution. The evolutionary algorithm is able to find significantly more feasible solutions compared to the state-of-the-art solution approach for a carousel-type location. The newly developed recombination technique in the EA delivers a promising performance for finding feasible shunting schedules which can help human planners work more efficiently.

Preface

This is the Master Thesis "The Application of a hybrid Evolutionary Algorithm to the Train Unit Shunting Problem.", a research which aims to find feasible solutions using a hybrid Evolutionary algorithm for TUSP. The research is conducted at the Research & Development department of Nederlandse Spoorwegen (NS). This Master thesis project is conducted as part of the Master thesis program Operations Management & Logistics at Eindhoven University of Technology.

The implementation of the Evolutionary Algorithm was difficult for me. Fortunately, Laurens, my first supervisor, and Roel, my company supervisor, have supported me throughout this project. I would like to thank Laurens specifically for the constructive feedback and support, and Roel for answering all my queries, especially during debug sessions.

Next to this, special thanks for Tim for providing feedback, despite the lengthy report. I would like to thank friends, family, and my girlfriend for the enjoyable moments during this period. Due to COVID I primarily worked in my student room, so going outside to see friends and family was important. In specific, I would like to express my gratitude towards my, parents Cees & Jacqueline, for the support over the years of education to get to this point.

Hopefully, you enjoy reading!

Eindhoven, 05 July 2022, Niek van Bavel

Contents

1 Introduction

The Nederlandse Spoorwegen (NS), or Dutch Railways, is the main railway operator in the Netherlands. To anticipate customer demand and commuter satisfaction, NS maintains a rolling stock. During morning and evening peak hours, the largest portion of the rolling stock is used to accommodate the flux of commuters. NS currently manages 757 train units of 7 different types of train units to accommodate a total of 246.531 seats. Additionally, NS will add 99 train units of a new train type to the rolling stock. Outside of rush hours, the demand for train units decreases enormously and excess train units are parked at shunting yards. The resulting surplus of trains is parked at the shunting yard off the main railway network.

The size and therefore capacity of shunting yards is limited. Not only potential extensions are limited, new optimal locations to create shunting yards are scarce and hard to find. Next to this, the continuous increase in the number of passengers results in an expansion of the aggregate number of trains. This combination of limited availability of 'parking space' and increasing number of trains creates a growing problem for shunting yards. NS aims to implement automated decision support, which can handle unforeseen changes. The shunting plans are designed by human planners, however, a local search method is being implemented by 2023 (Van Den Broek, Hoogeveen, Van Den Akker, & Huisman, 2021).

Shunting is the process of operating a shunting yard which includes three components: matching, parking, and routing. Matching entails the problem of assigning incoming trains to outgoing trains. The ordered sequence of train units is called a *train composition*. Furthermore, the problem of finding unobstructed movements for train units throughout the shunting yards required for parking or servicing activities is called routing. Parking is a sub-problem that entails finding a track for each train without exceeding the capacity of a track. The *Train Unit Shunting Problem (TUSP)* is the problem of finding an optimal shunting schedule for which every train departs on time and complies with the constraints of matching, parking, and routing.

Another important component which can be added are service tasks, which in combination with TUSP results in the train unit shunting and servicing problem (TUSS). The service tasks include maintenance, cleaning, and regular checks with a respective duration. These service tasks require a particular resource to execute the tasks at the shunting yard. This sub-problem aims to perform all service tasks before departure.

Once all constraints of these sub-problems are satisfied, a feasible solution is found. Hence, finding a conflict-free shunting plan is complex and time-consuming. The individual sub-problems are challenging, however, the strong dependency in TUSP result in an algorithmic complexity. The interaction of the components makes it practically impossible to effectively decompose the problem into smaller independent problems. Decomposition approaches do not seem effective for TUSP, which might be effective for similar complex problems.

NS is currently integrating a local search method to generate these plans automatically. The automatic decision support tools focus on helping human planners manage complex planning and scheduling problems at shunting yards. The local search method is the first algorithm capable of solving the complete Train Unit Shunting Problem for real-world scenarios(Van Den Broek et al., 2021). This is a significant improvement, however on some occasions, the method is not able to perform better compared to the human planners. Experiments show that the local search is not consistently able to find a feasible solution, while the human planners can find one.

Athmer (2021) introduced an Evolutionary Algorithm (EA) to increase efficiency for solving the TUSP. This EA integrates the local search algorithm to find feasible solutions. This research developed a problem specific conflict-based crossover to satisfy the constraints of TUSP. EAs were applied to various real world optimization problems (Jeong, Hasegawa, Shimoyama, & Obayashi, 2009), which perform well for global exploration, however, require more generations to converge to a feasible solution. The hybridization of EAs with local search methods can efficiently obtain highly accurate solutions (Moscato, n.d.; Yao & Mckay, 2001).

The evolutionary algorithm introduced by Athmer (2021) combined with a local search method does not include the complete TUSP. The evolutionary algorithm is not able to cope with splits and combines of train units, however to integrate an EA, this is essential to provide feasible and more optimal shunting schedules in real-world locations for NS. The arriving trains might require to be split or be recombined in order to more effectively supply the appropriate

departing compositions. The analysis of the evolutionary algorithm compared to the local search method reveals a varying performance on different kind of shunting yards. Several hypotheses aim to explain the fluctuating difference in effectiveness, however more research is required to give a definite answer.The main objective is to solve the TUSP using an evolutionary algorithm approach that integrates splits and combines of separate shunt trains. Next to this, this research aims to gain a deeper understanding of the performance for different type of shunting locations. Finally, the performance of finding feasible shunting schedules of the EA is compared to that of the current best local search algorithm.

1.1 Contribution of this research

The main contribution of this research is to present a hybrid EA capable of solving the Train Unit Shunting Problem including splitting and combining of train units. An essential part is a flexible adjustment of recombining the best elements of two parents into the offspring solution. The local search algorithm developed by Broek (2016) is integrated within the EA to increase the performance of the solutions. The proposed solution method is compared to the state-of-the-art local search algorithm, currently used by NS. The performance of these two algorithms are analyzed for the two different kind of shunting yards, namely a carousel and shuffleboard shunting yard. The ultimate objective of this research is encapsulated in the research question that states:

How to effectively configure an evolutionary algorithm to produce feasible solutions for the train unit shunting and servicing problem, taking into account splits and combines of train units?

An evolutionary algorithm consists of various elements based upon problem specific knowledge for it to work optimal. This configuration of elements is a decision which impacts the quality of the results. A hybrid evolutionary algorithm was able to find solutions for TUSP, however limited to scenarios without splits and combines procedure. The composition of arriving and departing trains is especially complex for the recombination element in the EA. First, a literature review is conducted to find current research about applying an EA to a constraint feasibility problem such as TUSP. Especially, it aims to gain a deeper understanding of establishing a recombination approach. This literature review aims to position this study amongst the current research about TUSP. The most important objective for this research is to identify the current research about a recombination procedure. The complexity lies with the combination of splitting and combining of train units that improve offspring solutions while satisfying the constraints of TUSP.

Can crossover operators be used to integrate splits and combines for tackling one or more sub problems of the TUSP?

The main contribution of this research is finding feasible solutions for TUSP scenarios including splits and combines. This is the first research that introduces an EA approach to apply to TUSP. The generic population-based meta-heuristic is inspired by biological evolution and performs procedures such as reproduction, mutation, recombination and selection to solutions in a generation. To optimize these solutions and explore the search space, an EA performs several procedures to the solutions of a generation. The procedures must satisfy hard constraints to generate solutions in a promising area of the search space. Hence, this project aims to find an application of the EA for scenarios including splits and combines. Therefore, the following sub-question is proposed:

How to implement splits and combines of train units in an evolutionary algorithm?

Finally, the performance of finding feasible solutions of the EA is evaluated for different types of locations. The shunting yards throughout the Netherlands can be categorized in: a shuffleboard, carousel and a mix of the former two. The performance of the EA is evaluated for an ideal shuffleboard and carousel location to gain a deeper understanding of the impact of the EA on finding feasible solutions. Finally the performance of the EA is compared to the currently best-performing solution approach of NS. The local search method is currently in use and developed for several years, hence the ideal performance measure. The performance of both solution approaches are evaluated using experiments at either an ideal shuffleboard and carousel location. Thus, resulting in the following sub-question:

In which circumstances is the evolutionary algorithm able to solve more problem instances than the current local search-based method?

In the remainder of this research, firstly an overview of recent literature on relevant shunting problems is provided in Literature Review. In addition, an overview of relevant contributions within the field of evolutionary computation is presented. This aims to summarize applications to a similar problem and distinguish a suitable recombination technique. Afterwards, Train Unit Shunting Problem provides a problem description of the Train Unit Shunting Problem. Evolutionary Algorithm approach to solve TUSP presents an overview of the elements of the evolutionary algorithm. An outline is presented of the recombination technique developed for TUSP. A setup of the experiments and the performance of the solution algorithm is provided in Results. Lastly, the conclusions and limitations of this research are presented in Conclusion.

2 Literature Review

Facets of this research for using an evolutionary algorithm for the TUSP are described in Introduction. TUSP is an important issue for NS and has been researched in the past. Various studies aimed to tackle TUSP using diversified approaches. The most substantial studies are presented within the first section. The next section provides an overview of applications of evolutionary algorithms for similar problems as TUSP. The overview assists the solution approach of integrating a split and combine procedure in an EA. The difficulty arise for selecting to generate new promising solutions by selecting the best-

2.1 Train Unit Shunting Problem Literature

This section provides an overview of current literature regarding TUSP. Earlier attempts to solve TUSP used mainly exact solution approaches. Afterwards, to include the full extent of the problem, heuristic approaches were developed. This section presents the overview of the development of the solution approaches for TUSP and the currently in-use local search algorithm.

2.2 Exact Solution Approaches

Freling, Lentink, Kroon, and Huisman (2005) introduced the train unit shunting problem (TUSP) which focused on two of the four sub-problems. It consisted of matching arriving train units to departing ones and parking these trains on a track at the shunting yard. The self-propelled trains can be coupled to to form any combination of train units. A new solution technique was developed in which the train unit matching is constructed using a decomposition approach. The authors used the standard Mixed Integer Programming (MIP) solver CPLEX to solve the mathematical model. Afterwards, a column generation approach is used to find a feasible solution for the parking sub-problem.

Lentink, Fioole, Kroon, and van 't Woudt (2006) uses a similar decomposition approach as Freling et al. (2005) to solve TUSP. In addition, they extended the TUSP by calculating the routing for each train movement. By composing the problem in four different steps and using a graph representation of the shunt yard in their study, they calculated the time required of moving a train from its arrival track to its destination track. Eventually, the routing problem is solved using the heuristic of the graph representation combining with the track occupation. Instead of solving TUSP step by step, Kroon, Lentink, and Schrijver (2006) constructed solutions by solving the matching and parking subproblem simultaneously. This increases the complexity of the problem resulting in an enormous amount of train collision constraints and eventually an immense computing time.

Furthermore, Haahr, Lusby, and Wagenaar (2015) presented a benchmark for multiple solution approaches for the matching and parking sub-problems of TUSP. The author presented a constraint programming model which assigns the adaptions to train unit compositions to tracks, instead of the events. This model required a high demand of memory, for which a delayed column generation heuristic is proposed. This was able to reduce the number of variables required by supplying a set of feasible solutions. Nevertheless, the reduction of computation time is exclusively reduced for less constrained models.

The exact solution approaches cope with the same difficulty, namely the computation time. Relaxation of the sub problems is required to reduce the computation time. To achieve this, sub-problems can be calculated separately or they are disregarded, which diminishes the quality of the solutions. These exact solution approaches are limited in performance at this point and not suitable for TUSP. Therefore, the next section will elaborate on heuristic solution approaches.

2.3 Heuristic solution approaches

The first heuristic approach is introduced by Haijema, Duin, and Van Dijk (2006) and is inspired by dynamic programming. It aims to maximize the grouping of train types on a shunting yard, minimize the disturbance risks and the amount of movements and (de)couplings to maximize the quality of the problem. The planning horizon is decomposed in smaller sub-problems which correspond to smaller planning periods to find good, robust and practical solutions. These smaller planning periods are solved using a heuristic which depend on formulated business rules. This dynamic programming heuristic could support the human planners by generating a set of plans. However, the heuristic is based upon business rules generated by human planners. Thus, generally the heuristic will not explore new search space unknown for the planners.

van den Akker et al. (2008) developed an integrated approach using a greedy heuristic and a dynamic programming algorithm. Matching rules plus track assignment are used in the heuristic to select the best action on arrival and departure. To reduce computation time, node pruning is used in the dynamic programming algorithm, however for larger instances it is not able to find feasible solutions.

Train unit shunt plans were developed by Jekkers (2009) using a genetic algorithm and the author introduced flexible shunt times for each shunt activity. The algorithm aimed to solve the parking and matching subproblem in order to tackle the shunting schedule to support local shunt planners. It used a genetic algorithm as search technique supported by a simple heuristic and a chromosome representation. The performance of a one-point, two-point, and uniform crossover was tested. For this research, the one and two point crossover performed best, whereas the uniform crossover performed worst. Eventually, the GA was able to perform better on larger problem instances than the MIP approach.

Jacobsen and Pisinger (2011) developed three meta-heuristic approaches to tackle the train parking and maintenance problem. A Guided local Search, Guided Fast Local Search and Simulated annealing resulted in significant less computation time compared to the MIP solver.

A stochastic local search was developed by Broek (2016) to find a global optimum which minimizes the objective function. The objective function aims to find a solution which solves all constraints of the sub problems, thus seeks to find a feasible solution. The local search algorithm attempts to minimize the objective function by shifting movements,modify parking locations, adjusting resource assignment and interchanging the assignment to positions of departing trains. The local search method, called Hybrid Integrated Planner (HIP), is able to adapt the routes using the shunt train activity graph (STAG), depicted in Introduction.

Peer, Menkovski, Zhang, and Lee (2019) developed a deep reinforcement learning solution by formulating the problem as a Markov Decision Process (MDP). The deep Q-network efficiently reduces the state space and develops an on-line strategy for the TUSP with the focus on dealing with uncertainty and delivering consistent solutions.

Various hybrid forms which use the local search method as a main component are introduced during the years. Van De Ven, Zhang, Lee, Eshuis, and Wilbik (2019) aimed to use machine learning in order to determine the capacity of shunting yards. This may decrease the search space and thus, increase the speed of finding shunting plans.The hybrid integrated planner performs well, however room for improvement remains.

de Oliveira da Costa et al. (2020) used a machine learning approach to improve the evaluation speed of the local search heuristic to determine if an instance has a feasible solution. The authors used a Deep Graph Convolutional Neural Network to predict feasibility and thus, whether to continue or abort the search process. This approach aims to improve the efficiency of the computation time to use on instances more likely to be feasible.

Eventually, Van Den Broek et al. (2021) succeeded to develop the first local search approach capable of constructing feasible plans for the shunting and service scheduling problem for real-world instances. The complete problem is described using partial order schedule representation. The proposed solution is capable of solving shunting problems and excels compared to other approaches. Further research is done to improve the robustness of the model to complications of actual operation and to extend the scope of the model.

van der Linden et al. (2021) developed a train unit shunting and servicing simulator demonstration track which provides the user with a state and all feasible actions. After an action is picked, TORS calculates the result and the process repeats.

One valuable contribution is the evolutionary algorithm introduced by Athmer (2021) in combination with the local search method developed by Van Den Broek et al. (2021). The author used the initial solution and representation of the local search. The most promising contribution was the Conflict-Based Crossover (CBC) designed for this EA. This approach performs a crossover which results in a high probability of improving the solution, more details about the crossover are presented in Memetic Algorithms. Furthermore, after performing the crossover, it executes a local search in order to improve the offspring to a local optimum. Several different parameter settings for the evolutionary algorithm were tested compared to the HIP on an artificial location, a carousel location, and a shuffleboard location. The artificial location is a rather simple shunting yard whereas the others are real world locations. The performance of the algorithms were evaluated by Athmer (2021) using the percentage of problem instances solved and averages time in order to present the result of the experiments. The evolutionary algorithm did not perform well for De Grote Binckhorst and the artificial location whereas the local search performed much better. However, the EA with CBC outperformed HIP on the Kleine Binckhorst. The artificial location and de Grote Binckhorst are categorized as shuffleboard locations, whereas de Kleine Binckhorst is classified as a carousel location. The main difference within the composition of tracks is a single entry/exit and two entries/exits, respectively. The author did hypothesize the underlying reason for the fluctuating results. One reason could be the size of the search space for different locations. In a shuffleboard location, a larger portion of the shunting yard is occupied which results in less flexibility and a smaller search space, which benefits the local search. A carousel location has a larger search space which makes the crossover of an evolutionary algorithm more effective. Another theory is that the Grote Brinckhorst has a larger number of trains within the instances, which might result in a higher probability of infeasible solutions using crossover. Experiments should verify the above-given hypotheses. The research by Athmer (2021) does not integrate splits and combines which is essential to use the EA in a real-world setting.

For some years now, studies tried to tackle the TUSP using different kind of approaches, which presents us with some insights. Due to the complexity of the problem, research demonstrated that heuristic approaches outperform exact techniques on TUSP instances. The amount of constraints and computation time gets enormous to solve an instance. Furthermore, the most promising research at this point is arguably the local search method of Van Den Broek et al. (2021). This approach is able to solve all four sub-problems simultaneously. Furthermore, various research is conducted to explore the complete search space by integrating the local search algorithm in a hybrid solution approach. This observation is transferred in this research by combining a computational intelligence approach with the local search method.

2.4 Computational Intelligence

The previous section presented the difficulties of TUSP and solving them using different solution approaches. The research combines an evolutionary approach with the local search algorithm to find feasible solutions for TUSP. Previous research has established some elements for an EA to tackle TUSP. However, literature is absent for an application of an EA to scenarios with the option to split and combine train units. This research review aims to explore current research for finding a similar technique to address this problem. This section presents the most relevant applications of some of the principles of computational intelligence. These are nature inspired solution approaches that contain similar characteristics.

Computational Intelligence is application of biologically and linguistically inspired computational paradigms. The field covers adaptive mechanisms which enable intelligent behavior in dynamic environments. The following three main paradigms of computational intelligence will be shortly described here, namely: artificial neural networks, fuzzy logic and evolutionary computation.Five main paradigms which are considered as a part of computational intelligence are artificial neural networks, fuzzy systems, swarm intelligence,

2.4.1 Evolutionary Computing

Evolutionary computing is a sub field of computer science, inspired by the process of natural evolution A. E. A. Eiben and Smith (2003). The ability of diverse species tailored to survive within their own niche embodies the power of

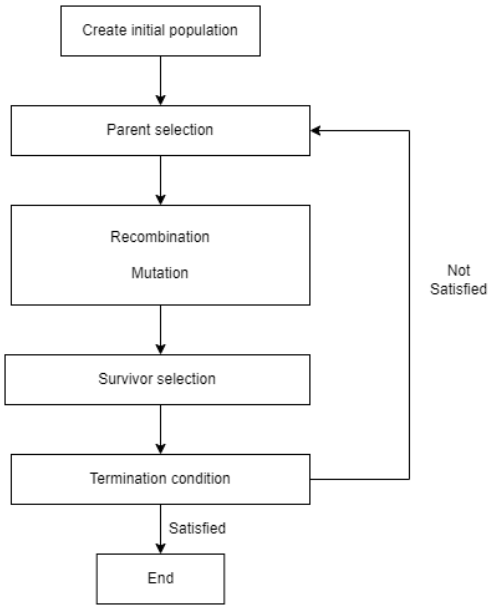


Figure 1: A general overview of an EA

evolution. The fundamental metaphor for this distinct style of problem solving relates to trial-and-error, or generate-and-test. A generation of individuals aim for survival and reproduction within a given environment. A natural variety of characteristics exist within the individuals of the population. The environment determines the ability of each individual to fit to the conditions and thus, the chances of survival and multiplying. Individuals better able to adapt to the conditions will survive and produce new generations. Reproduction and mutation will result in a gradual evolving population and increase the fitness. Meanwhile, in the context of problem-solving, a collection of candidate solutions aim to solve a problem. The quality of each solution is the ability to solve the problem of each candidate. This determines the chance that the solution will be used for construction of new candidate solutions.

Inspiration for EA/ brief history The application of Darwinian principles for problem solving is found as early as 1940s. The first computer experiments were executed by Bremermann in 1962 in: "Optimization through evolution and recombination". The Evolutionary computing, first introduced by Holland (1992b) is the application of Darwinian principles within computer science research. Evolutionary programming and genetic algorithms developed separately and eventually became one technology called evolutionary computing. It uses the concept of survival of the fittest, the individual that is best able to adapt to the environmental conditions (A. E. A. Eiben & Smith, 2003). Evolutionary computation is the domain which includes the following main algorithms: Evolution Strategy, Genetic algorithm, Differential Evolution, Genetic programming and evolutionary programming (Slowik & Kwasnicka, 2020). A wide variety of industrial applications can be addressed by varieties of these techniques. Slowik and Kwasnicka (2020) presents many applications using each of the techniques, however the authors highlight the discussion, which technique suits for a particular problem. It suggest that no given technique suits a particular problem, however, the better one understands a method the better the application can be fine-tuned.

Overview of an Evolutionary Algorithm This section EAs are discussed in more detail. Various components and procedures are applied to define a particular EA. Many variants and extensions of an EA exist which will be explained in section . Nonetheless, a general overview of an evolutionary algorithm is presented in figure .

Representation The first step in defining an EA is to translate the original problem context to the problem-solving space. This might involve a somewhat simplified or abstract representation of the problem to create a well-defined problem. It is important to consider how a problem should be specified and manipulated by a computer. Phenotypes are the possible solutions in the original problem context while the encoded individuals are referred to as genotypes.

The initial population is evaluated to determine the performance of each solution. The evaluation function is the foundation for selection. The fitness function defines the requirements the population should adapt to. It is a function which assigns a quality measure to genotypes.

The function of a population is to hold possible solutions. An individual itself does not change, solutions change over generations. The population is typically defined by the number of individuals in it. A supplementary feature, a distance measure or neighbourhood relation, can be used to determine the extent populations evolve within the context. The difference in solutions present within a population is referred to as diversity. It can either be defined as the number of different fitness values present, the number of different phenotypes or genotypes.

Parent Selection Mechanism Parent selection creates a distinction of individuals based upon their quality. The better adapting individuals become parents of the next generation. A parent is an individual which is selected to perform variation procedures to produce offspring. Typically, parent selection is probabilistic which allows high-quality individuals to have a higher chance to produce offspring.

Gomes, Farias, and De Magalhães (2018) evaluated the performance of several parent selection methods in a niching genetic algorithm: most similar and most dissimilar and binary tournament selection compared to random parent selection. For this problem, the most similar parent selection procedure resulted in the best overall results. Whereas the most dissimilar parent selection strategy resulted in the worst performance of the methods.

Variation Operators Variation operators evolve solutions and create the necessary diversity and thereby enable novelty. This procedure generates new solutions and aim to improve them. Two different types of operators exists: *Recombination* which is an *n-ary* operator and *Mutation*, which is an *unary* operator.

*Recombination The procedure of creating a new individual solution using the information from two (or more) parent solutions is called recombination, also referred to as crossover. Combining two partial solutions using recombination is a method that characterizes EAs. The probability of recombination operators are determined using a crossover rate, P_c. Traditionally, two parents are selected which create two offspring with probability P_c. The function of a recombination of genetic material is to ensure search progress and explore new promising regions of the search space. The information exchange assures that offspring obtains genetic information from both parents. Thus, features of the best performing individuals can be inherited by the offspring.

Various new recombination techniques are developed over time, Andreica, Chira, Andreica, and Chira (2015) made a summation of the performance of various crossover for permutation based encoding. The crossovers discussed are: Order crossover 1 + 2, cycle crossover, partially-mapped crossover, One point crossover, Two-point crossover, uniform crossover, position based crossover. The study proposed Best order crossover, in which parent 1, parent 2 and the global best are included.

Andreica et al. (2015) demonstrated a summation of various crossovers for permutation based encoding. The crossover discussed are: Order crossover 1 + 2, cycle crossover, partially-mapped crossover, One point crossover, Two-point crossover, uniform crossover, position based crossover. The study proposed Best order crossover, in which parent 1, parent 2 and the global best are included.

Ray, Kang, and Chye (n.d.) presented an EA for constrained optimization problems. To enhance solution compatibility the algorithm establishes non-dominance solutions independently in the constraint and objective space. The recombination technique yields three additional solutions, generated by a uniform crossover and random mix and move. A constraint matrix was introduced, which indicates the degree of satisfaction per solution. Furthermore, This matrix was incorporated with the objective matrix as a single aggregate measure. Afterwards, a non-overlapping constraint satisfaction technique was applied to the solutions. This technique establishes a mating pair if the constraint satisfaction complements the other solution. Eventually, the approach aims to generate solutions with better constraint satisfaction using a set of rules to determine the mating pair. The results indicate a significant decrease in iterations to obtain comparable objective function values.

The effectiveness of a recombination operator is heavily dependent on the problem. To find a suitable operator the researcher should investigate the problem. Sharma, Blank, Deb, and Panigrahi (2021) defines two approaches to find the best operator for a particular problem. It either relies on problem dependent knowledge and select the operator according the EA literature for a similar application or by imitating the past study operators. More applications of

recombination techniques within EAs are presented in Evolutionary Computing

Described in Train Unit Shunting Problem, the representation used in the EA is a STAG representation. To develop a hybrid evolutionary algorithm, this should also be capable of processing this type of solutions. This is especially eminent in the reproduction procedure. To execute a crossover or mutation, it is best if the procedure is able to reproduce while keeping the sub problems in regard. The aim of reproduction procedure is to discover the search space, however using an improper reproduction operation, the algorithm will steer towards infeasible solutions. Rothlauf (n.d.) Binary, integer or real-valued string representations can make use of standard recombination and mutation operators. This makes it easy to predict the performance of the evolutionary algorithm by using existing studies. If the representation of problems whose phenotypes are not binary, integer, or real-valued, it becomes more complex. The standard mutation or recombination techniques can not be used. In such case, specialized operators are necessary to inherit the valuable properties form their parents (Radcliffe & Surry, 1994a).

This research searched for similar mechanisms as the splits and combine procedure in TUSP for an EA. Some of the reviewed fields for similar research is in swarm robotics Nagavalli, Chakraborty, and Sycara (2017), modular autonomous vehicles Lim et al. (2018), and multi modal combined transportation Li et al. (2021). The multi modal combined transportation could overlap in some ways, if multiple vehicles are reassigned at a place to other various vehicles. No relevant research is discovered which could be applied for this study. In addition, the constraints of TUSP increase the complexity of integrating a split and combine procedure.

*Mutation Mutation gives a variation to exclusively one parent and creates one child. It applies a mutation rate, P_m, to use a randomised change to the representation. The purpose of mutation is to introduce diversity into the population. It aims to prevent local minima by avoiding all individuals in a generation becoming too similar to each other. Cicirello (2006) evaluated the performance of a swap mutation and insert mutation. These operators operate as follow: it randomly selects two positions and swaps them, the other removes a location and insert it in another randomly chosen location. Mutation does not converge as fast as recombination techniques. However, it is able to focus the search at a near potential optimal region by creating offspring similar to the parent.

Survivor Selection Mechanism The offspring selection determines which individuals remain in the next generation. Usually, the population size is constant which indicates that a set of original and new individuals are discarded. The fitness value evaluates the individuals and concludes which individuals survive. A commonly used mechanism is the tournament selection. Herein, the fitness value of *n* number of individuals are evaluated and the best individual is selected to survive.

Termination Condition Finally, the performance of every generation is evaluated to decide whether the iterative algorithm should continue or terminate. The ideal situation for an EA is to find the optimum or a feasible solution and thus, terminate. EAs are a heuristic algorithm so there is no guarantee of finding the optimum or feasible solution. Termination criteria are utilized to prevent it from running infinitely. Some criteria commonly used exclusively or in combination are:

1. Maximum allowed CPU time.
2. Threshold of the predefined number of evaluations
3. Minimum level of diversity within a population
4. A required value of improvement per generation.

2.4.2 Fuzzy Logic

Fuzzy logic is the reasoning of approximation rather than resulting in a Boolean outcome. It is inspired by the processes of human perception and cognition and based upon the idea of relative graded membership. Fuzzy logic processes information based upon computational perception, namely vague, imprecise, without sharp boundaries or partially true information. Fuzzy logic is used in many applications which are amongst others: washing machines, vacuum cleaners, stock trading or weather forecasting systems (Singh et al., 2013).

Fuzzy logic does not implement a recombination technique in the same way a genetic algorithm does. However, a hybrid form of genetic algorithms and fuzzy logic is applied to real world problems. Lee and Takagi (1993) proposed an automatic fuzzy system design method which uses a genetic algorithm. The genetic algorithm determines the number of fuzzy rules and membership function shape and position and consequent parameters simultaneously as input for the fuzzy system.

In contrary, Jalali Varnamkhasti, Lee, Abu Bakar, and Leong (2012) proposed a genetic algorithm which depends on the output of the fuzzy system. The performance of a genetic algorithm is highly dependent on genetic operators and specifically the crossover operator. A fuzzy logic controller determines the crossover operator and probability selection based upon the diversity within the population. Consequently, a set of fuzzy rules control the crossover operator and its probability. The results indicate an effective method to find high quality solutions.

Likewise, Im and Lee (2008) presented an adaptive crossover, mutation and selection technique using a fuzzy system for a Genetic Algorithm. The fuzzy systems allows performance improvement by implementing problem specific information in the genetic operators. The adaptive crossover and mutation probability performed much faster than using fixed values. Hajri, Liouane, Hammadi, and Borne (2000) used a controlled genetic algorithm based on a fuzzy system to solve the NP-complete job-shop scheduling problem. Once the elite approaches the desired solution, the crossover probability is greater than the mutation probability and vice versa. The controlled genetic algorithm showed robustness against variation and flexibility between the number of generations and the desired solution.

Typical CSPs adopt a static partition for solving constraints into strong and weak constraints. (Guan & Friedrich, 1993) applied a fuzzy set theory to constraint satisfaction which specifies the degree a constraint is satisfied. Fuzzy Constraint Satisfaction Problem (FCSP) acknowledges the domains of variables and constraints to be fuzzy. λ is the satisfaction index which can be exploited to find better or even the optimal solution. In extension, Kowalczyk (1998) presented a genetic algorithm to solve FCSP. The objective function of the GA is the satisfaction degree of fuzzy constraint specified by the FCSP. The potential solutions are evaluated to satisfy the fuzzy constraints to the maximum degree.

2.4.3 Neural Networks

Artificial neural networks (ANNs) were originally proposed by McCulloch and Pitts (1943). The development of ANNs is heavily inspired by the characteristic functioning of the human brain. The biological and artificial networks have the building blocks in common, simple computational units which are highly interconnected. next to this, the function of the network is determined by the connections between the neurons. A distinction in two main categories of neural networks is made by Lek and Park (2008):

1. Supervised Learning
   The ANN learns the level of performance and correct behavior in the learning phase.
2. Unsupervised Learning
   The ANN autonomously analyzes properties of a particular data set and learns to reflect the features in the output.

Further explanation and literature review of artificial neural networks can be found in

One combination of evolutionary algorithms and neural networks is the hybrid multi-objective evolutionary design for artificial neural networks. Yao (1999) suggested a GA which evolves the weights, architectures, learning rules and input features of an ANN. The combination result in a powerful framework which is able to adapt to a dynamic environment. The NeuroEvolution refers to the optimiziation of ANN using evolutionary computation algorithms and are applied in various applications (Papavasileiou, Cornelis, & Jansen, 2020).

Bull (1997) proposed a neural network trained on individuals to generate an explicit fitness function. The resulting model of the fitness function is processed by the EA to find a solution. This architecture optimizes a problem which is hard to express mathematically or difficult to simulate. This technique is shown to be effective for a wide range of function types.

A combination of artificial neural networks combined with reinforcement learning is applied to TUSP. Peer, Zhang, Menkovski, Lee, and Huisman (2018) developed a deep reinforcement learning solution by formulating TUSP as a Markov Decision Process. This approach is able to efficiently reduce the state space while capable of dealing with uncertainty.

Most of the applications of neural networks are within optimization problems. However, this solution approach is in addition to optimization problems suitable for constraints satisfaction problems, like TUSP. According to Bourret and Gaspin (2003), a neural network, for which it is possible to design by using the problem representation by a constraint programming language, can solve any CSP. It shows how an a constraint problem can be transformed to a neural network which solves the identical problem. For large size problems, it mixes constraints programming and neural networks. Finally, it presents an application of the method to solve the tasks assignment problem.

de Oliveira da Costa et al. (2019) introduced machine learning approach to complement the local search and accelerate the search process. The evaluation speed of the local search is improved significantly based upon the number of shunting yards and size of the planning problems. Furthermore, it introduced a deep graph convolutional neural network to predict the feasibility of TUSP solutions obtained during the LS. This research combines machine learning to improve the performance of the local search of finding feasible solutions for TUSP.

2.5 Evolutionary Computation

The TUSP is a complex and interdependent problem as described in Train Unit Shunting Problem. The previous section gives a broad framework of applications of some of the many computational intelligence approaches. General approaches were presented to approach constraint satisfaction problems and NP-hard problems. This section presents an overview of the variations within evolutionary computation. The variations are inspired by the same principles of natural evolution. However, the areas within evolutionary computation were developed for a different initial application area which resulted into different philosophical frameworks.

Evolutionary computation is a broad research field, this research considers the approaches which are suitable to handle the sub-problems of TUSP. For example, Evolution Strategies is a strata which pursues the natural evolution processes for seeking procedures. An individual is represented by three vectors, namely: the set of object variables, the set of step sizes and the set of inclination angles (Hoffmeister & Bäck, 1990). This problem solving heuristic is unsuitable to handle the STAG representation of TUSP and thus not dealt in more detail within this section. This section aims to find a recombination technique applied to a similar problem as TUSP or a similar technique applied in another research field. The methods considered in this section are genetic algorithms, memetic algorithms, differential evolution, evolutionary programming and genetic programming.

2.5.1 Genetic Algorithms

The most generally known evolutionary algorithm is the genetic algorithm, Initially conceived by Holland (Holland, 1992a). A classical genetic algorithm includes a binary representation, a one-point crossover, a bit flip mutation, a fitness proportional parent selection and a generational parent selection, such as elitism (Goldberg, 1989). Traditionally, Genetic algorithms pursue a fixed workflow. First of all, a certain population of individuals fill an intermediary population using parent selection. The intermediary individuals are shuffled to create random pairs and a crossover is applied to each pair with probability p_c. The new generation replaces the parents immediately and undergo mutation. Each individual solution is modified with an independent probability of p_m. The next generation exists of the resulting individuals.

The vehicle routing problem is a combinatorial optimization and integer programming problem which aims to optimize a fleet of vehicles to distribute in order to deliver to a set of customers. The vehicle routing problem is commonly tackled using genetic algorithms. The straightforward presentation of a genetic algorithm enables various crossover operators. The 1 point crossover, just mentioned chooses a single point in the array and takes the first part from one individual and the rest is taken from the other individual. (Karakatič, 2021). Other fairly straightforward crossovers

Algorithm 1 Pseudocode of Genetic algorithm

```
t = 0;
initialize (P(t = 0));
evaluate (P(t = 0));
while isNotTerminated () do
  P_p(t) = P(t).selectParents ();
  P_c(t) = reproduction (P_p);
  mutate (P_c(t));
  evaluate (P_c(t));
  P(t + 1) = buildNextGenerationFrom (P_c(t), P(t));
  t = t + 1;
end
```

which work with path representation are ordered crossover(Prins, 2004), standard two-point crossover (Bean, 1994) and exchange mutation, which exchanges two genes (Karakatič & Podgorelec, 2015).

For a certain problem, the type of crossover is adjusted to give the right fit. Yueqin, Jinfeng, Fu, and Jing (2007) developed a crossover operator which randomly dispatches a customer in the individual with poorest quality. The chromosome with poorest quality is destroyed, afterwards it is repaired by connecting the previous customer of the removed one and the hinder one in the shortest way possible. Whitley (2001) proposed an edge recombination operator. Traditional optimization methods were not as effective or applicable to sequencing problems. The authors developed a genetic operator which avoids cities without a continuing edge. These isolated cities have a higher probability of being chosen next and connected to the edge with fewest connections. These depict crossover operators in a more complex environment which solve a problem ad can be of great use in our search for a relevant crossover.

Rojas, Montero, and Riff (2017) presented a Greedy Randomized Adaptive Search Procedure (GRASP) based approach for solving the departure matching problem. The authors implement a constructive and a local search step which deal with the constraints and costs of the given problem. The GRASP algorithm uses a pre-processing, construction and post-processing phase. The pre-processing phase is an initial solution of matching incoming and departing trains. The construction phase determines which incoming trains are possibilities to be matched to departing trains. Afterwards, a train is randomly selected from the generated candidate list and added to the solution. Than, a local search phase using a hill-climbing algorithm is performed.

The Hunter Valley coal chain is a complex supply chain in Australia. To address the train scheduling problem of the rail infrastructure, Sanhueza, Mendes, Jackson, and Clement (2020) developed an improved genetic algorithm. A train from the train fleet is chosen based on rail network, jobs and available train fleet, routing are not part of the problem definition. The objective is to minimize the total travel times and is subject to five constraints. An array of integers mapping the train's velocity represents a solution. The research performed a uniform crossover strategy for the recombination of two solutions and a 5% probability of mutation. The genetic algorithm performed well and generated slightly better than the solutions of the schedulers.

According to Thierens (2010), parts of two good solutions can be used to juxtapose by crossover to generate new good solutions. This is one option to benefit the search efficiency by a search bias of recombination operators. The alternative is to shield the partial structures of two good solutions of variation disruption so the new solution inherits the part. The first option could only work if recombination does not disrupt the partial structures too often. Designing the solution representation or crossover operators in an appropriate way can achieve this. This knowledge can also be induced from a population of solutions if there is an inadequate of domain knowledge to design such solution or operator. Linkage learning is the field of learning what variables compose of important partial solutions and should therefore be protected from disruption (Chen & Lim, 2008).

Thierens (2010) developed an alternative linkage learning called the Linkage Tree genetic Algorithm (LTGA). A hierarchical clustering algorithm produces the linkage tree which is constructed for each generation and acquires linkages between the problem variables. To produce new offspring, the LTGA selects two parent solutions and visits the linkage tree from the root to the leaf. A crossover mask is defined by the clustering at the particular tree node which determines the recombination at each branching point. The LTGA shows it can solve hard functions efficiently by traversing the linkage tree for each tree node.

Hu and Di Paolo (2008) proposed a GA to solve the airport gate assignment, a NP-hard problem. The success of implementation a GA relies on the degree of linkage in building blocks within chromosomes. The linkage information in a chromosomes in the GA are the absolute positions of an aircraft in the queues to gates and the relative position between aircrafts. This method identifies and protects good linkages which result in a good performance during comparative simulation studies.

Multi-objective evolutionary algorithms aim to find Pareto-optimal solutions simultaneously rather than a single optimal solution. A critical distance between single-objective and multi-objective optimization is that the latter one include a multi-dimensional space, on top of the usual decision variable space (Deb, 2001). The multi-objective optimization search identifies two goals:

1. Identify a set of Pareto-optimal solutions.
2. Identify a diverse set of solutions that represent the entire range of the Pareto-optimal front.

The different solutions may display trade-offs, the user can evaluate the solutions often based on qualitative and experience-driven information.

Schaffer (1985) proposed a Vector Evaluated Genetic Algorithm (VEGA) to generate multiple solutions for multi-objective problems. The objective of VEGA was to identify multiple classification rules to cover the complete set. The set covering problem was solved by choosing a fraction of the next generation based on one of the attributes. VEGA was able to generate multiple solutions, however VEGA generated exclusively extreme solutions on the front. Horn, Nafpliotis, and Goldberg (1994) introduces the non-domination ranking and selection for converging populations towards the Pareto-optimal front. Furthermore, the authors suggested some kind of niching to avoid converging to a single point. The niched Pareto GA performed well, however was very sensitive to the configuration of several parameters.

To overcome the weaknesses of earlier research, non-dominated sorting genetic algorithm (NSGA) first ranks the order of non-dominance and afterwards uses a surrogate fitness function (Bagchi, 1999). It was one of the first evolutionary algorithms which focused on moving toward the true Pareto-optima region. However, criticism remained about the approach (Deb, Pratap, Agarwal, & Meyarivan, 2002):

1. Computational complexity
   The non-dominated sorting algorithm used in NSGA includes a computational complexity of $O(MN^3)$. The number of objectives is specified by M and N specifies the population size. Every generation is involved in the non-dominated sorting procedure which causes the large complexity.
2. Lack of elitism
   Elitism can speed up the search for good solutions significantly and it is able to maintain them (Zitzler, Deb, & Thiele, 2000).
3. The sharing parameter σ_{Share} For this genetic algorithm, it is required to specify the sharing parameter which is not desirable.

Deb et al. (2002) developed a non-dominated sorting-based genetic algorithm II which tackles all three difficulties described above. The authors were able to reduce the computational complexity to $O(MN^2)$. Furthermore, a new selection operator was designed to introduce elitism. The NSGA-II procedure is presented in Figure 2. The current population of the t^{th} generation is population $R_t = P_t \cup Q_t$ of size 2N. The members of population R_t are sorted

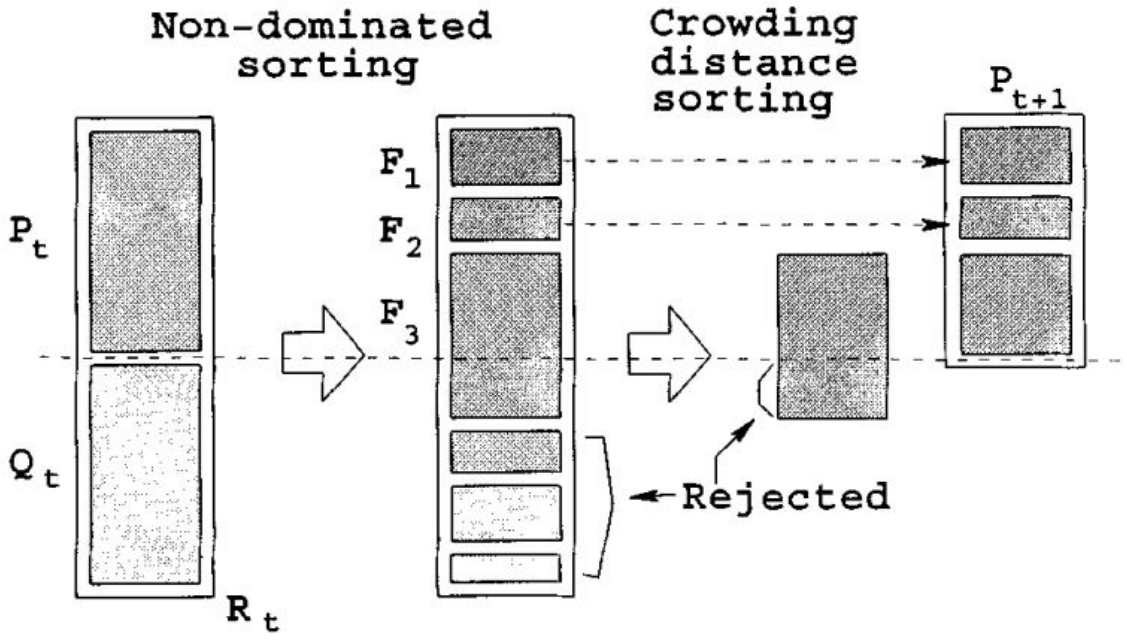


Figure 2: NSGA-II procedure by Deb et al. (2002)

using non-domination. The best performing solutions belong to set F_1in the population. If the number of members in F_1 is smaller than that of N, the remaining members are chosen from subsequent non-dominated sets, in this case F_2

The NSGA-II procedure selects the solutions belonging to the best non-dominated set, depicted in 2 as F_1. If F_1 is smaller than N, all members of the set are chosen for the new population represented as P_{t+1}. Afterwards, the solutions from subsequent non-dominated fronts are chosen, thus F_2 and eventually F_3.

Ponsich, Jaimes, and Coello (2013) provided a survey on multi-objective EA applications for the solution of portfolio optimization problems. It presented a large amount of portfolio optimization problems by model development and high ability of adaption of MOEAs to new models.

Dozier, Bowen, and Bahler (1995) developed a micro-evolutionary hybrid solution to solve randomly generated constraint satisfaction problems. This Hybrid search technqiues uses a hill climbing in combination with an evolutionary search to speedely find solutions for Constraint satisfaction problems. After evaluation, it exceeds performance of the Iterative Descent Method (IDM) (Morris, 1993). Another evolving algorithm to solve constraint satisfaction problems is developed by Bain, Thornton, and Sattar (2004). The difficulties addressed are related to the matching algorithms to specific constraint satisfaction problems. It uses a representation which fits to either complete or local search heuristics. Empirical study show that the new framework is able to evolve algorithms to solve the boolean satisfiability testing.

2.5.2 Memetic Algorithms

The word 'meme' was first used in 1976 in the book 'The Selfish Gene' and defined as 'the basic unit of cultural transmission or imitation' Ong, Lim, and Chen (2010). Memetic algorithms are a hybridization of at least two heterogeneous entities either through a natural transformation or conscious manipulation. The aim is to combine two or more methods to maximize problem-solving capabilities, pseudocode of a standard memetic algorithm is illustrated in algorithm 2. Memetic algorithms are intrinsically concerned with exploiting all available knowledge of the concerning problem. A fundamental feature which characterizes memetic algorithms is the incorporation of problem

domain knowledge (Moscato & Cotta, 2006). Information is not transmitted unaltered by communicating parts, which is illustrated by the word "Memetic". Instead, information is processed and enhanced between individuals. This is achieved in memetic algorithms by integrating heuristics, local search approaches, specialized recombination operators, approximation algorithms and more. A generation of agents cooperate and compete within the search strategy to optimize the population (Norman & Moscato, 1991).

Algorithm 2 Pseudocode of a Memetic Algorithm

```
t = 0;
initialize (P(t = 0));
P(t = 0) · localSearch();
evaluate (P(t = 0));
while isNotTerminated() do
  P(t) = selectIndividuals ();
  mutate (P(t));
  P(t) · localSearch();
  evaluate (P(t));
  P(t + 1) = buildNextGenerationFrom (P(t));
  t = t + 1;
end
```

Amaya, Porras, and Fernández Leiva (2015) demonstrated the local operator implementation and combination in a memetic algorithm. This research introduced a synergistic combination of general-purpose methods and problem specific add-ons. The research aimed to introduce problem knowledge in the optimization technique. The research established the use of global optimization (population-based) and local(trajectory based) techniques.

Bärecke and Detyniecki (2007) developed a memetic algorithm based on permutation enconding for matching two inexact (sub-)graphs. This is a NP-complete problem for which the memetic algorithm provides fast solutions. Crossover operators preserving relative position or order do not fit to the graph matching problem. By suggesting a 5 step problem specific position-based crossover (PBX) operator it sustains solely absolute positions. The memetic algorithm nearly always converges to the global optimum for large instances in a considerable shorter amount of time.

Moscato and Cotta (2010) defined the recombination operator as a process in which a set of parents is manipulated to recreate a set of descendants. The new configurations rely on the identification and combination of features extracted from the parents. Three properties of interest can be exhibited by recombination operators (Radcliffe & Surry, 1994b):

1. Respect
   This represents the exploitation component of a recombination procedure. A recombination operator is considered as respectful, if, it generates offspring carrying all main characteristics common to all parents.
2. Assortment
   The exploratory component of recombination is represented by assortment. Recombination is considered to be properly assorting, if, it configures offspring which carry any combination of compatible features from the parents. The assortment is considered weak if multiple recombinations are required before achieving this effect.
3. Transmission
   The transmission represents the intuitive role of recombination. A recombination operator is regarded to be transmitting if every feature of the resulting offspring is present in at least one of the parents. Thus, no new information is introduced, as this performed by the mutation operator.

These three properties aim to describe the input and output behaviour of a recombination procedure. It characterizes the potential offspring which can be produced by a particular operator.

To some extent, the real-world train timetabling problem approached by Semet and Schoenauer (2005) is similar to TUSP. It deals with the tightly constrained combinatorial problem of construction of train schedules by adapting departure and arrival times and allocation of resources. For this problem a memetic permutation-based evolutionary algorithm is developed. In contrast to TUSP, it aims to minimise the total accumulated delay by satisfying constraints for arrival and departure times, stopping time, speed, safety spacing, connections and switching gates. The hybridised algorithm performs promising for large, complex instances and quickly converges to a satisfactory zone of solutions.

The extended job shop scheduling problem for multiple factories includes high interdependencies of production schedules for individual factories. Shiroma and Niemeyer (1998) designed an EA, successively, a Tabu search is applied to reduce the number of evaluations required. A non-binary chromosome representation is introduced to efficiently process large complex data sets. Applying standard genetic crossover and mutation operators would produce many invalid solutions. As a result, problem constraints are reflected in special domain-dependent variation methods to guarantee validity of newly generated individuals. A variation of Davis (1985) OX crossover operator is used to ensure that all jobs are exclusively once present in the resulting solution. The best performance was achieved by including the domain-dependent information along the combination of an EA and Tabu search.

A relevant hybrid EA is the research of Athmer (2021), introduced in Literature Review. This hybrid EA processes the STAG representation of solutions and tackles TUSP. The research introduces a conflict based crossover which is based upon problem specific knowledge. This is required to improve the probability of improving the offspring solutions.

2.5.3 Differential Evolution

Storn and Price (1997) proposed differential evolution on a broad variety of applications. Just like other variants within evolutionary algorithms, it is a population-based optimizer which tackles the initial point problem by maximizng a particular objective function at global optimizer. Differential evolution is applicable to continuous optimization problems which include a real value objective function.

Differential evolution utilizes either discrete or continuous recombination. Discrete recombination is an operation which copies trial vector parameters from randomly selected vectors. These crossovers can be applied to binary, real-valued and symbolic data. Continuous recombination uses a linear combination of vectors as trial vectors and thus, can not be applied to symbolic data or binary variables. Discrete and continuous recombination techniques have a variety of implementations to differential evolution, in more detail described in Price, Storn, and Lampinen (2005) .

C. Zhang, Yang, and Li (2018) presents an interesting approach of evolutionary computation as the main difference is the way which individuals are made up. Scatter search is used to improve the distribution of the solutions. This is an evolutionary technique which systematically constructs new solutions by combining existing ones. The strategy uses search diversification and intensification which resulted in a high performance for a variety of optimization problems. For this two way motor train scheduling problem, a matrix of 6 by 6 is used for the scatter search. The trail vector of discrete differential evolution is generated by cross operator. The trial vector is acquired by the combination of the mutation vector and the target vector. The authors designed differential variation operators for integer permutation in discrete differential evolution. The hybrid discrete differential evolution algorithm was able to obtain promising results.

An original algebraic approach for differential evolution algorithms is proposed by Santucci, Baioletti, and Milani (2016) for combinatorial search spaces. The concept of a finitely generated group is exploited and specialized for permutations space. This discrete DE algorithm is applied to the permutation flowshop scheduling problem with the total flowtime as objective. Furthermore, a novel selection strategy, a crossover operator for permutations, a heuristic-based initialization and a memetic restart procedure are proposed in the study. The differential evolution algorithm was able to find some new best known results.

Storn and Price (1997)This discrete differential evolution algorithm uses 4 priority index crossover algorithm to order the individuals and avoid breaking the constraint relationship between genes and destroy individual characteristics, hindering the evolution process of the population.

A new design of a differential evolution is proposed by Ali, Essam, and Kasmarik (2020) which is suited to tackle

the complex multicomponents travelling thief problem, which is a combination of the travelling salesman problem and the knapsack problem. The proposed method includes mapping and repairing methods, a modified mutation operator and a local search method. The interdependencies of the sub-problems increases the complexity for generating solutions. The research showed that finding the optimum for both sub-problems, would not guarantee an optimum for the travelling thief problem. complexity of solving sub-problems simultaneously can be recognized in the sub-problems of TUSP.

2.5.4 Evolutionary Programming

The original application of evolutionary programming lies in predicting the next state in a sequence of symbols in a finite-state machine (L. J. Fogel, 1964). A suitable representation of a finite-state machine specifies initial state, the total number of states, input-output symbols pairs and the next-state transition of each of the states. According to Hinterding (2000), crossover or recombination procedure was not used, as the complex representation was highly unlikely to produce a solution which represents a valid finite state machine. The mutation operator was used as most of the nature of the parent is still represented in the child, it obtained better results than combining it with recombination (D. Fogel, 1992).

2.5.5 Genetic programming

The field in Evolutionary Computation for evolving computer programs is called Genetic Programming (GP). Generally, the explorative operator used in GP is the common and simple one point crossover. However, the reliance of the crossover technique on random selection and placement is accepted as the restricting factor of the performance (Yuen, 2004). The crossover performs damaging crossover in useful sub trees and ignores the context of a sub-tree-to-be-exchanged. The success of a recombination operation depends on the placement of a sub-tree in the right framework is essential (Majeed & Ryan, 2006),

Majeed and Ryan (2007) developed a new crossover operator named context-aware crossover to avoid issues of the standard recombination operator. The operator works by evaluating the randomly selected subtree from one parent and inserting it in every possible location in the second parent. The best possible context is found by evaluating the resulting solutions and finally selecting the best child from the pool of all options. Evaluating all options increases the computational time, however the drastic increase of performance permits the use of significant smaller generations. The context-aware crossover is tested the 11-bit multiplexer problem, which the standard GP is unable to solve. Due to the constructiveness of the operator, the context-aware crossover is able to handle such complex problems and perform well on a wide range of problems.

Compared to other evolutionary computation techniques, genetic programming includes a complex representation of individuals. In Geometric Semantic Crossover, offspring is created by a convex combination of two parental individuals. In other words, offspring is produced at the internally dividing points of the semantic vectors of parents (Hara, Kushida, Tanemura, & Takahama, 2016).

2.6 Constraint handling of EAs

Typical search operator, recombination and mutation in evolutionary algorithms, are unsuited to handle constraints. If parents satisfy certain constraints, the offspring might violate them (A. E. Eiben, 2001). EAs are able to handle constraints directly, indirectly or a mix of those.

2.6.1 Direct Constraint Handling

Handling constraints directly entails if violated, it does not affect the fitness function. This implies that no bias towards solutions satisfying them exists and thus, the generation will not become more feasible concerning these constraints.

A disadvantage of them is that they are heavily problem dependent, however, if implemented, they generally work well (A. E. Eiben, 2001). Common procedures of direct constraint handling are:

1. Eliminate infeasible solutions.
   The probability of creating a feasible solution in the first generation is practically zero. This approach would eliminate all solutions in a CSP, and thus not effective in these circumstances.

2. Repair infeasible solutions.
   To repair infeasible solutions, a repair procedure is required to modify an infeasible solution into a feasible one (Coello, n.d.). Developing a repair procedure is highly problem specific.

3. Preserve feasibility by special operators.
   This technique aims to preserve feasibility across generations using special problem dependent operators. The offspring remains within the feasible search space, if the parents were feasible. Athmer (2021) developed a conflict based crossover which preserves feasibility of solutions for TUSP, more details can be found in Evolutionary Computing.

4. Decode.
   Decoding transforms the search space which is different from the search space in the original problem. The feasible solutions are produced by using elements of the new search space as input for the decoding procedure. This produces feasible solutions and a space where *free* search can be performed (A. E. Eiben, 2001).

2.6.2 Indirect Constraint Handling

Indirect constraint handling is typically viewed as penalties for constraint violation in the optimization objectives. Every violation of a constraint is given a certain penalty value which quantifies the severity of the violation. The objective function represents the amount of violation, if it is zero, a feasible solution is found. The objective function is defined in the local search and used in the EA.

Comparing evolutionary algorithms on binary constraint satisfaction problems.

2.6.3 Repair Algorithms

Various studies develop a problem specific recombination procedure to generate feasible solutions for permutation representations of tours of the traveling salesman problem. Hoos and Stützle (2005) use a standard recombination operator instead and performs a repair mechanism to restore valid tours from infeasible paths. The repair mechanism focuses to preserve as many edges of the current infeasible edges as possible. The infeasible edges are replaced by an edge closest to it to make the path more feasible. This technique presents another way to generate feasible solutions in an efficient way.

The multidimensional 0-1 Knapsack problem (MDKP) is studied using a Quantum Inspried Evolutionary Algorithm (QIEA) by Jindal and Bansal (2019). Exact approaches are not suitable for large number of items and dimensions due to execution times and excessive space requirements (Akçay, Li, & Xu, 2004). The generated population by QIEA presumably produces solutions which do not satisfy dimensional constraints. The repair algorithm ensures that the constraints are satisfied and aims to maintain and possibly improve the fitness value. The authors evaluated the performance of four repair algorithms namely: Simple repair, random repair, sorted repair and a combination of multiple repair approaches.

The simple repair approach basically removes items from the individual one by one until the constraints are satisfied. Afterwards, the knapsack is filled up again by adding items. With respect to the second approach, an item is randomly removed and afterwards, the items are randomly added back. The sorted repair algorithm first sorts the items based on ascending order of weights. Afterwards, it removes items until the point that the constraints are satisfied and finally, the removed items are added back in opposite order. The last algorithm uses k number of sorted repair and afterwards

the $k + 1$ generation uses a sorted repair, in Jindal and Bansal (2019) $k = 5$. The simple linear repair approach results in a less effective approach compared to others. The other approaches perform very well, for which the sorted repair converges faster in terms of required time.

X. Zhang, Ma, Ding, Fang, and Qian (2022) introduces a co-evolutionary algorithm based upon the auxiliary population. This paper proposes a constrained large-scale multi-objective supply chain network optimization model concerning multiple objectives and constraints. First, one population is used to solve the problem without any constraints. Afterwards, a repair operator is used to improve the initial infeasible solutions. The solution algorithm is able to outperform other compared algorithms, particularly for large-scale instances.

The multi-task allocation model for mobile crowdsensing aims to find the optimal participants-task pair using limited resources. Ji, Guo, Gong, and Shen (2021) introduces an evolutionary algorithm with problem-specific repair strategy. This aims to generate superior feasible solutions , the new genetic operators are presented as a problem-solver. This research presents a problem-specific repair strategy, specifically suited for the constraints of the multi-task allocation model.

2.7 Conclusion

Current research about TUSP indicates that exact solutions, concerning computational time, are not suitable at this point. Various studies initiated a solution approach, mostly for some of the sub-problems. Eventually, a local search algorithm was able to result in a very promising performance. This research tackles the complete problem and can find feasible solutions for real-world instances in a limited amount of time.

First, the section presents an overview of applications of studies using a solution approach of fuzzy logic systems or neural networks. These studies demonstrated that these meta-heuristics result in good-performing solution approaches for complex NP-hard constraint problems.

Furthermore, the section presents an overview of studies that apply an EA to a complex NP-hard constraint feasibility problem. Especially hybrid evolutionary approaches demonstrate promising performance for a complex problem. The combination of the explorative aspect of an EA and the exploitative character of the local search results in a solid solution approach. An EA can encounter difficulties to converge to the exact solution, which is the strength of a local search algorithm. Furthermore, the elements of an EA are very flexible to fit the problem. In this way, constraints can be satisfied in each phase of the EA. Additionally, constraints can be handled in two ways, direct or indirect. The direct constraints must always be satisfied whereas the indirect are the constraints to be optimized over generations.

Different techniques are applied to satisfy the constraints of a problem. One way is to repair solutions to meet the hard constraints. However, as TUSP is a highly interconnected problem, it can get very complex to repair the generated solutions. A constraint satisfaction technique that suits TUSP is using direct and indirect constraints. However, all components of an EA must satisfy these constraints to find feasible solutions. Various studies showed the adjustment of variation operators to the instance-specific knowledge. In this way, the direct constraints are satisfied while allowing improvement to find a feasible solution for the indirect constraints. This approach will be used to satisfy the constraints of TUSP and include a split and combine procedure in the EA. This literature review provides an overview of the most relevant studies. No other studies have introduced an recombination technique to satisfy the constraints of TUSP and include a split and combine procedure in an EA. This is objective for this research.

3 Train Unit Shunting Problem

The previous section provides us with an overview of the literature about TUSP. The approach for this research is a hybrid evolutionary algorithm to solve real-world cases of TUSP. The aim is to increase the performance of finding feasible solutions by combining an evolutionary algorithm approach with the local search introduced by Van Den Broek et al. (2021). This section provides a detailed overview of the facets of the TUSP problem and corresponding difficulties for a hybrid EA to solve it.

3.1 Problem Description

The first section provides the preliminaries of a shunting yard location. Afterward, the sub-problems within the TUSP problem are presented. Furthermore, the required details of the problem are presented. These are the details that are required to understand the general TUSP. Next, a more detailed description is given of the local search algorithm by Broek (2016) for TUSP, since this will be integrated into the evolutionary algorithm. Next to that, the EA presented by Athmer (2021) is discussed as this is used as a framework for this research. Afterwards, the currently best-performing local search algorithm is describe in-use local search algorithm is described and the

3.1.1 Preliminaries

A shunting schedule establishes a shunting plan which incorporates all features of the Train Unit Shunting Problem. The schedule provides a shunting plan that all trains are well-maintained and clean before their projected departure. A feasible schedule covers all sub-problems of TUSP. This section provides a detailed overview of the sub-problems of TUSP. ‘ The infrastructure of a shunting yard consists of a set of tracks, connected by a set of switches. Each track has a length that determines the capacity. Tracks are distinctive based on which side they are accessible. Figure 3 presents the two types of tracks, a LIFO-track and a free track. A free track is approachable from both sides, hence both train units are able to depart at any point. The LIFO-track is connected to a bumper, and can only be accessed from side A. Train unit 2 is not able to depart before train unit 1.

Figure 3: The Free track (left) and LIFO-track (right) are depicted by the solid lane, the dotted lanes represent connections to adjacent tracks and 1 and 2 represent parked train units.

Most shunting yards consist of a combination of LIFO- and free tracks. Two main types of shunting yards are distinguished; the carousel structure and shuffleboard structure. A carousel structure shunting yard mainly exists of free tracks, which enable a carousel-like movement. The shuffleboard structure consists mostly of LIFO-tracks. NS manages the ’Grote Binckhorst’ which is a large shuffleboard location and the ’Kleine Binckhorst’ which is a carousel-type location. The number of tracks for these shunting yards vary from 9 to 24.

The incoming trains can be split and recombined to form the outgoing train compositions. The interim compositions are referred to as *shunt trains*. Each train unit is part of a shunting train and can not be part of two shunt trains simultaneously. The combination of train units is only possible if the train types are corresponding.

Trains that require service require a duration and a specific location. To perform a service task, a certain facility is required that is located along a track, such as a cleaning platform. Reversing the direction of a train is called a *saw*

move. The control of a train is transferred from one end to the other, in general these are time-consuming operations. The *gateway-track* is the track at which shunt trains arrive and depart a shunting yard.

3.2 The Sub-Problems of TUSP

The train unit shunting problem is defined using four sub-problems. Lentink et al. (2006) introduced the four sub-problems namely: matching, parking, routing, and cleaning. In addition, details for a split and combine operation are provided. This section provides a detailed overview of these different components of TUSP.

3.2.1 Matching

The matching problem consists of the assignment of a set of incoming trains to the set of outgoing trains. A train arrives at a specific time and consists of one or more train units. Likewise, an outgoing train consists of one or more train units and departs at a particular time. The focus of the matching problem is to match each arriving train unit to a specific position in one of the departing trains, such that the train enters the service site before departure and the train sub-types match.

If coupled train units arrive and are assigned either to different departing trains or in a different order, the train units are split on the service site. Likewise, if train units are assigned to the same departed train and not (correctly) coupled at arrival, a combine operation is required. Split and combines are often necessary at the shunting yard. However, the number of composition modifications is minimized due to the time required for split and combine operations.

3.2.2 Parking

The main objective of the parking problem is twofold: to prevent the parked trains from exceeding the track length at any point and, to ensure that each shunt train is able to have an unobstructed path if it departs from its track. A shunt train that arrives at a track joins the trains already parked on the track. The track is only accessible from one side if it concerns a LIFO-track. The parking problem is highly influenced by the configuration of the service site. The order of arrival is important for departures, and the type of track it is parked on.

It is not allowed to park, split, and recombine on *gateway-tracks*. The gateway track is the track at which shunt trains arrive. Shunt trains require a parking location at another track to execute split and recombine procedures.

3.2.3 Routing

The objective of the routing problem is to acquire an unobstructed path for each movement while minimizing the movement duration. The movement duration is estimated based on the number of tracks, switches, and saw moves required to obtain the pathway. If a moving train is blocked by one or more trains, it is defined as a *crossing*. On the occasion that shunting train unit 2 attempts to depart the LIFO-track before train unit 1 in figure 3, a crossing occurs. A shunt plan including a crossing results in an infeasible schedule. However, due to safety regulations, simultaneous movements are not allowed which significantly simplifies the routing problem since intersections are impossible.

3.2.4 Servicing

A predetermined set of service tasks determines the requirement for each train unit. To provide each service task, limited resources are required for its duration. Service tasks do not have a precedence relation. The aim of the task scheduling problem is to complete a schedule which accommodates all service tasks and all trains can depart in time.

3.2.5 Splitting and Combining

Splitting and combining of train units have no distinct neighborhood search in the local search approach. The local search, although capable of configuring any combination, splits and combines at most only once. If necessary, the split occurs at the first track the train unit moves to after arrival and the recombine at the last track before departure. An arriving train is divided precisely in the configuration determined by matching and based upon length restrictions. To reduce the size of the search space of the local search algorithm, the division and combination of train units are exclusively allowed immediately after arrival or just before departure. This assumption allows the algorithm to focus on the other sub-problems of TUSP.

The shunting routing problem is comparable to a 6x6 rush hour puzzle. Hauptman, Elyasaf, Sipper, and Karmon (2009) successfully implemented two evolutionary algorithms to solve these puzzles. The shunting maintenance problem can be regarded as an open shop scheduling problem with machine flexibility, buffer and blocking constraints, deadlines and release dates. G. H. Kim and Lee (1995) introduced Evolutionary Intracell Scheduler which was applied to open-shop scheduling problems. Combining all sub problems and solving them simultaneously results in a strong dependency between the components. Adjusting a parking location might result in new routing conflicts. Therefore, it is impossible to solve the sub-problems sequential using the researches described above.

Broek (2016) introduced a solution approach to handle all sub-problems of TUSP. This local search is able to initiate solutions and adjust to provide solutions to help human planners, more details are added below. The objective is to find a feasible solutions, satisfying all constraints of each sub-problem.

3.3 Constraint Satisfaction Problem

Heuristics are commonly applied to optimisation problems, whereas TUSP problem is a *Constraint Satisfaction Problem* (CSP). This is a class of models which represent models that include a set of variables and a set of constraints. The form a A. E. Eiben (2001) defined the CSP as a pair (S, ϕ) with S as a free search space and ϕ is a Boolean function of S. A valid solution for a CSP is when finding an $s \in S$ with $\phi(s)$ = True. Furthermore, the authors define a recombination or mutation operator *free* with respect to the search space if the operator does not exceed the search space. Common crossover operators from genetic algorithms are free with respect to the search space. The highly interdependent sub-problems do not create a *free* search space so restrictions to the search operators are required to keep offspring within boundaries.

Problems from different application domains can be translated to a CSP. Amongst others is the N-queens problem (Haralick & Elliott, n.d.), temporal reasoning Tsang (1987), scheduling (Dincbas, Simonis, ECAI, & 1988, n.d.) and line labelling in vision (Waltz, 1975). Furthermore, CSPs are a subset of the NP-complete problems (Garey & Johnson, 1979).

3.4 Hybrid Integrated Planner

The hybrid integrated planner is currently the first method to solve real-world problem instances of the complete shunting and scheduling problem Van Den Broek et al. (2021). This research presents a partial order schedule representation that captures all the sub-problemm of TUSP. First of all, it computes an initial solution using a simple sequential algorithm. Afterwards, it selects candidate solutions from the neighbourhood for the next iteration of the search process. The simulated annealing algorithm randomly selects a neighbor and accepts it as the candidate solution for the next iteration. Furthermore, Broek (2016) introduced a representation technique, objective function, and local search operators for TUSP which are described in this section.

3.4.1 Evaluation of solutions

For each generated solution, the corresponding fitness value of each solution is calculated. A solution is considered feasible, once all constraints described in Train Unit Shunting Problem are satisfied. To accomplish feasible solutions, direct and indirect constraints are utilized in this *Constraint Satisfaction Problem*. The local search algorithm implements the indirect constraints to determine the fitness of generated solutions. A solution must consistently satisfy these direct constraints:

- A service resource does not exceed the limit of one task on any occasion.
- Simultaneously movements are never allowed.
- The matching of incoming and outgoing train units is correct.
- The sequence of movements and tracks for train units is correct.

The additional constraints are introduced as indirect constraints in the objective function. These properties are optimized to obtain feasible solutions:

- The number of active and passive crossings, i.e. collisions.
- The number of occasions that the total length of occupying trains exceed the track limit.
- the delay of departure trains.
- The delay of arrival trains. This occasion only occurs if a train arrives concurrently with a movement of another train, as simultaneously movements are not allowed.
- The number of departing trains that require a combination operation on its gateway track.

Each of these violations is multiplied by its corresponding weight. The severity of a penalty is determined by the weight, which ultimately determines which solution is closer to feasibility. All penalties cumulative is the fitness of a solution, it is considered feasible if this equals zero. Eventually, this research aims to benchmark its performance with HIP. The quality of solutions of HIP is analyzed using these features. Thus, an excellent way to benchmark the algorithms is to use the same feasibility criteria.

3.4.2 Solution Representation

The local search algorithm requires an appropriate solution representation. The Shunt Train Activity Graph (STAG) was first introduced by Broek (2016) and captures the important aspects of a solution while allowing easy modifications.

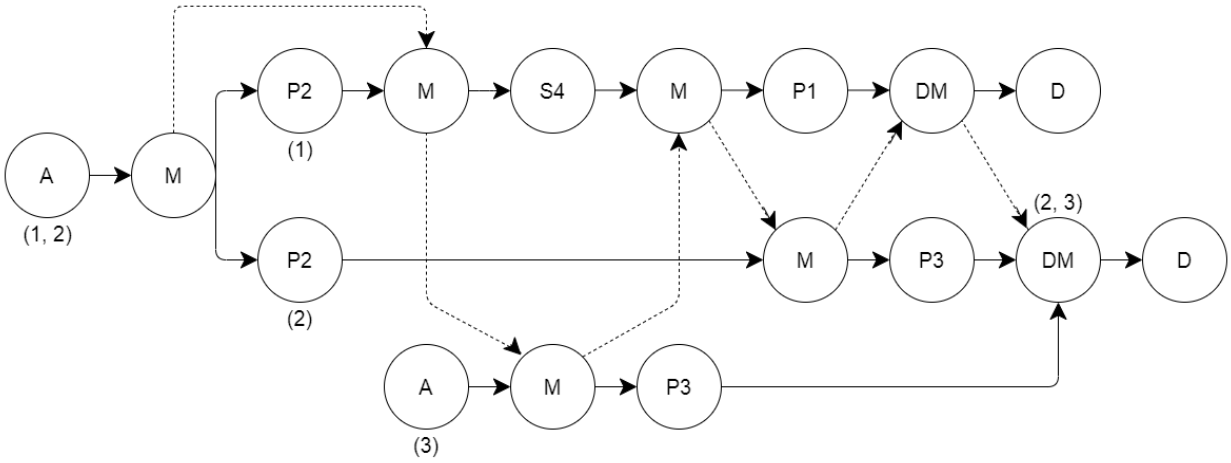


Figure 4: Shunting train activity graph:

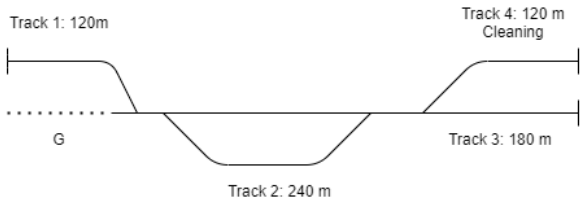


Figure 5: Example service site

The following example visualizes an activity graph for a scenario with 3 incoming train units and 2 departing trains. Figure 5 shows a small example shunting yard including the arrival track G and four tracks with the corresponding length on which train units can be parked. For this service site, it is only possible to perform the cleaning service task on track 4. Furthermore, tracks 1, 3 and 4 are required to leave in Last In, First Out (LIFO) order, which makes it only possible to arrive and depart from a single direction. Track 2 is accessible from both sides and thus train units are able to leave in both directions, providing more flexibility. There are two arriving and two departing trains which are presented in the timetable in Table 2. Next to this, the required service tasks for each train unit is presented in 1.

Train Units	Type	Service Tasks
1	ICM-3	cleaning (34 min)
2	ICM-3	none)
3	ICM-4	none

Table 1: The train units at the example scenario.

Arriving train	Time	Departing Train	Time
(1,2)	10:00	ICM-3	10:45
(3)	10:30	ICM-4, ICM-3	12:00

Table 2: Arriving and departing trains for the example scenario.

The corresponding shunting train activity graph for the given scenario is presented in Figure 4. Various node types are distinguished based upon their activity namely:

1. The Arrival (A) nodes correspond to the information on the arrival time and track. This activity represents a train movement to the main railway network.
2. Likewise, the Departure (D) nodes correspond to the outgoing train and incorporate departing time and track.
3. The Parking (P) node represents shunt trains parking at the location.
4. Service tasks (S) represents the service required by a train movement, i.e. cleaning or maintenance and contains the track.

The nodes described above contain the information of the activities at the shunting yard and are referred to as *track activity nodes*. Each of these nodes contains both information about the track and the side of the track. The track activity nodes have at most one incoming and one outgoing arc. Movement nodes are capable to connect the activities at the shunting yard which are:

1. The Movement (M) nodes contain the information of the sequence of tracks and connect all activities at the shunting yard.
2. The Departure Movement (DM) is the concluding movement and always precedes the departure node.

The movement nodes are connected by at least one incoming and one outgoing arc. The occasion that more than one incoming arc is present, indicates a combination operation. If there are multiple outgoing arcs from a movement node, it represents a split operation. The difference in the train composition of the shunt trains implicitly represents the splitting and combining of train units. HIP assumes that combine operations happen directly after arrival if required. Similarly, combine operations occur just before departure from a track.

A STAG representation explicitly models the movement and service orders. Next to this, the matching of the solution follows from the shunt train distribution at the departure nodes. However, the activity graph does not explicitly include the paths taken by train movements and the start time of any activity. To keep track of the routing, the local search algorithm generates an activity graph that orders the activities in the movement activity set, such that there are no simultaneous movements.

The computation of routing and time assignment is twofold. First of all, the total order in the activity graph is used to compute a path for each movement activity. The number of crossings and track capacity violations are computed using this path. All movements are ordered sequentially and therefore the routing problem can be formulated as a single-source shortest path problem. Based on the total order of the movements, the routes can be accurately calculated. Eventually, the informed search algorithm A* Hart, Nilsson, and Raphael (1968) is used to find the shortest path.

The second step is the assignment of the start times to all activities in the activity set. Since the train movements are scheduled sequentially, the start time is computed as the maximum end time of all predecessor nodes in the activity graph. The addition of the times finalizes the schedule and now it could be used as a service site schedule.

3.4.3 Initial Solution

The first step for an EA is the solution representation. The shunt train activity graph, presented in section 5 is applied. This provides an accessible overview of a solution while incorporating the essential details. The STAG representation is developed to present TUSP and therefore is used as the representation for the EA.

The local search method provides an initial solution by a simple sequential algorithm for the TUSP. Firstly, a perfect matching between arriving and departing trains is constructed which solves the first sub-problem. This procedure ensures that no incoming train unit is matched to an outgoing train unit before successfully executing all service tasks.

The train unit matching indicates the minimum number of splits and combines required to transform the arriving train units into the appropriate departing trains.

The initial solution maintains combined train units in place if it satisfies three conditions, namely:

- The train units are assigned on a departing train in the identical order.
- The arrival time and the total of the service tasks do not surpass the departure time.
- Each service task can be executed without exceeding the track limitations.

Afterwards, based on this outcome, the arrival and departure nodes are added to the solution to initialize the partial order schedule. The arrival and departure nodes are connected using movements concerning the splitting, combining, and matching determined in the previous procedure. Next, the service activities are arranged based on increasing departure time, resulting in a service schedule. The first service task in order is assigned to the first service resource available. The task is assigned to the resource with the smallest total workload, otherwise, it is randomly selected. The service activities are added to the solution, exclusively after the train units split and ahead of combining, if applicable. To attain service tasks, movements toward and forward are added. Lastly, parking locations are assigned to the trains with an adequate capacity between consecutive movements.

3.5 Local search

The local search aims to improve the initial solutions by selecting new candidate solutions. The candidate solutions are adjusted to maintain a correct solution regarding the sub-problems of TUSP. The search operators either change locations in the existing solution or adjust the activity graph directly by adding a movement or activity. The local search operators available to adjust existing plans are:

1. **Track reassignment** This operator adjusts the track on which a train is parked in a shunting plan. To maintain a correct plan, the movement arriving at this parking location and departing from it is also adjusted to the new parking location.

2. **Shift movement**
   The rearrangement of movements in the partial order schedule is referred to as a shift movement.

3. **Insert movement**
   This operation corresponds with the insertion of an additional movement since some occasions might benefit if a train temporarily parks at a different location.

4. **remove movement**
   A redundant train movement can be removed by the local search.

5. **Service order swap**
   HIP searches for valid solutions which can be established by swapping the same service resource or by swapping the tasks for the same shunt train.

6. **Resource reassignment**
   A service task is assigned to a different suitable resource.

7. **Matching swap**
   This operator search adjusts the matching of incoming trains to outgoing compositions.

All of the local search operators respect the hard constraints of TUSP. The objective of the local search is to change existing solutions and to compose a more feasible solution. HIP gradually improves the candidate solutions by integrating small changes. HIP is currently the best-performing solution approach for TUSP.

3.6 Evolutionary Algorithm to TUSP

Athmer (2021) attempts to enhance the performance by introducing a hybrid evolutionary algorithm. On some occasions, HIP was not able to find a feasible solution whereas the human planners were. A hybrid EA has the advantage over a local search algorithm to explore the search space more effectively. This research introduced a diversity metric to quantify the diversity within a generation. The most important contribution is the conflict-based crossover. This crossover operator aims to select the best performing elements of each parent solution to establish offspring in a promising area of the search space while satisfying the TUSP constraints.

3.6.1 Selection of Parent Solutions

First of all, solutions are selected which are determined by parents. Diversity is used to determine which parents will be combined next. Each solution is transformed from a STAG representation into a string representation. Then, the diversity is calculated for each pair of solutions. This determines the amount of alternations required to convert one string in the other. The individuals which require the most adjustments to be identical are the ones who are selected for crossover. This aims to ensure that the parents are as diversified as possible and therefore the full potential of the search space is explored.

3.6.2 Selection of Survivors

The initial solutions are used as input for the evolutionary algorithm. Once they are generated, the parents for the next generation are selected. A constant population size n is maintained throughout generations. To ensure this $2n$ solutions are produced before selection. After which, a procedure called tournament selection is performed to recruit solutions.

Tournament selection randomly divides $2n$ individuals into groups of a predetermined size. This can be adjusted to the preferred size, in this research, $n = 2$). Afterwards, the tournament selection benchmarks the solution concerning the fitness value. The best performing, thus lowest fitness value, is selected and included in the n solutions. The next procedures of the algorithm are performed on these solutions.

3.6.3 Conflict-based selection of Trains

The pair of parents are determined using the diversity measure. Afterwards, the best-performing elements of the parents' solutions are determined. The components of a solution are picked based on the lowest amount of conflicts evoked in a solution. Thus, the conflicts are calculated for each train and, if favored, this sequence of tracks for this particular train is his implies that the sequence of tracks is appointed to the solution.

To determine the best performing train of a parent, the Conflict based crossover handles three criteria:

1. **The amount of conflicts**
   The objective function used in the local search method measures the fitness of solutions. A conflict is an event that increases the objective value, for example, the crossing of train units. The number of conflicts in each train unit is calculated. The ones with the fewest conflicts are chosen from the parent solution as it is most likely

that this is a valuable part of the parent solution. The train units involved in the same number of conflicts will remain.

2. **The neighbour count within the shunting plan**
   Two trains parked on the same track at the same time are defined as neighbours. The total amount of neighbours that a specific train unit possesses is calculated from the current plan which is selected from the parents. The neighbour count assures train units with zero conflicts remain together, thus preserving a good order. The neighbour count is applied as follows:

   (a) The train unit from the parent with the highest neighbour count is selected if the conflict cost for a unit is zero.

   (b) If the conflict cost is greater than zero, the parent with the lowest neighbour count is selected.

3. **The overlap of train units on a track** Overlap is defined as two train units that park at the same track at some point in time. The train unit which has the least overlap with already selected train units is selected.

Athmer (2021) evaluated the performance of the EA including the conflict-based crossover. The solution approach resulted in promising results for carousel locations, however, HIP unquestionably outperformed the EA for shuffleboard locations. The EA is not able to handle splits and combines in the crossover, which limits the usability tremendously. Figure 4 presents a small scenario in which train units 1 and 2 split on track 2 and eventually train units 2 and 3 recombine at track 3, indicated by the numbers in parenthesis. This simple scenario requires a split and a combine operation to achieve a feasible shunt plan. The amount of required splits and combines increases for a higher number of arriving and departing train units. Due to the strong dependencies among the sub problems for the train unit shunting problem, adjusting just one parking track could affect various movements within a Shunting plan. Therefore, performing a crossover on two parents with distinct matching and different splits and combining results most likely in an invalid solution.

TUSP is an extremely complex problem due to the dependency of the sub-problems. Solving the sub-problems sequentially does not provide feasible solutions, so it is required to solve them simultaneously. HIP introduced a solution representation to capture the essential details and allow easy modifications. Next to this, this section provides an overview of the local search algorithm introduced by Broek (2016) which produced solutions to solve the complete TUSP. Afterward, the core concepts of the hybrid evolutionary algorithm developed by Athmer (2021) is provided.

4 Evolutionary Algorithm approach to solve TUSP

The inter-dependency within the TUSP results in a high number of constraints. Section 2 presents an overview of the many applications of an EA for complex feasibility problems. An evolutionary algorithm can be applied to various real-life problems and obtain excellent performance. The algorithm includes several components that are configured to fit the particular problem. Section 3 presents an overview of the relevant research of TUSP. Elements of HIP and the hybrid EA are used to define parts of this research.

This section centers its attention on the composition of the instruments in an evolutionary algorithm. The EA is flexible to a high degree as the components can be adjusted to fit a particular problem. First of all, an overview is presented of the general EA. Afterwards, the main contribution of this research is demonstrated which aims to recombine two parent solutions in one offspring solution, while satisfying the constraints of TUSP.

4.1 Components of the Evolutionary Algorithm

Figure 6 presents an overview of the evolutionary algorithm with a local search integrated used in this research. This EA is a combination of elements from literature and from this research. The elements used from literature are described in Train Unit Shunting Problem. An EA is flexible and all elements can be adjusted to the specific problem to meet problem constraints. HIP provides initial solutions which adhere the hard constraints described in Train Unit Shunting Problem. The crossover and local search will evolve the initial solutions quickly so a simple sequential algorithm can be used. Next to this, the objective value defined by HIP is used for this EA. This determines the quality of the solutions in a generation.

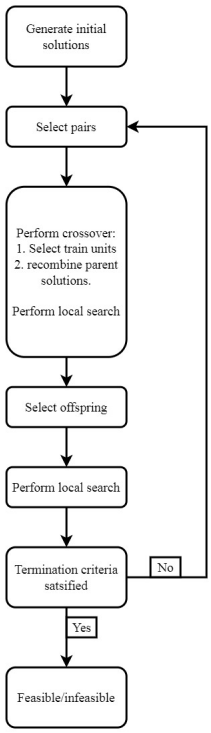


Figure 6: An overview of all elements of the hybrid EA.

Athmer (2021) introduced a diversity based metric to select parent pairs. Next to this, it introduced the conflict-based crossover to conserve the best performing elements of each parent in the offspring. These elements are used in the hybrid EA of this research. The adjustments required for the EA to cope with splits and combines procedures are introduced in this research. Furthermore, the recombination of two parents into one new solution is introduced in the next section.

4.2 Recombination Technique

The main contribution of this research lies within the recombination of different solutions. Fusing the best-performing elements of two solutions is complex. To ensure a correct solution, all criteria need to be met. The recombination technique exists in different parts which focus on preserving the optimal performing components while guaranteeing legitimate solutions.

The number of conflicts, neighbour count, and overlap described in Conflict-based selection of Trains is calculated for all individual train units. To ensure a correct graph for train units separating after arrival, an entire train is selected. Most locations have an arrival track in which splitting and recombining is not possible. If two train units of an arriving train are selected from a different parent which has dissimilar arriving tracks, it causes a complication. The train units are not able to split before the movement towards the track. To prevent this from happening, all train units assigned to a single train are selected from that parent, visualized in figure 7 for train units 2602,2401.

4.3 Splits and Combine Procedure

The main contribution of this research is the recombination of the best-performing train units into the new offspring solution. No other studies have studied TUSP using a hybrid evolutionary algorithm for scenarios including splits and combines. First of all, the conflict-based crossover selects train units for the offspring. Afterwards, the best parts of the individuals are combined into their offspring. Due to the splits and combines, the individual train units require some adjustments to result in a correct plan graph. Asserting a correct plan graph includes correct train positions for each movement and thus, track and movement activity is evaluated. Conflicts on the other hand, are the irregularities within the solution that the algorithm can process and aims to solve ultimately.

In the crossover, two parents exist which recombine to recreate offspring. The parents are evaluated for their correctness and thus, the track and movement activity is certainly legitimate. The conflict-based crossover determined the trains which performed the best and so as little as possible is changed from this. First of all, the matching is consistently taken from parent A. This ensures that the departure of each train is possible. Next to this, the service tasks in the EA are duplicated from the parent. This ensures a correct schedule for the new offspring solution. The local search operators can improve the matching and servicing in a solution.

Splitting is executed if required, on the track right after arrival, recombination is performed on the track before the departure movement. The departure movement is the movement sequence required for the train to exit the shunting yard. Thus, train units are separated at the first parking location available, whereas train units are joined at the last parking location available before the departure movement.

To ensure correct separation and recombination of train units, an identical parking location is essential. The location of separation for train units is guaranteed as this parking location is unaltered to that of the parent individual by selecting trains instead of train units. The recombination of train units is accomplished using a similar method. recombining train units is achieved on the last parking location before departure from parent A. The parent of the solution guarantees that this combination is possible and thus, can be used for the new solution. Eventually, the movement towards the parking location, the parking location, the movement from the parking location towards the departure movement, and the location of the departure movement are altered to connect the train units correctly.

Example

The next scenario illustrates how these alternations are realized. Table 3 presents an overview of the incoming trains, the train types, and departing trains for this instance. The incoming trains and train type is fixed, whereas the particular train for the departing trains is flexible. This scenario includes five train units, which are three different types of trains assigned to three incoming and outgoing trains.

Incoming Trains	Train Type	Departing trains
2603	SLT-6	SLT-6, SLT-4
2401, 2602	SLT-4, SLT-6	SLT-6
2601, 2402	SLT-6, SLT-4	SLT6-SLT-4

Table 3: The available incoming and required departing trains for this scenario.

Table 4 provides an overview of the available material for this scenario. Additionally, it specifies the corresponding train length and preconditioned service tasks. Figure 5 represents the shunting yard that accommodates the train units. This is an ideal four-track carousel location including one incoming and outgoing track. A more detailed description of the mechanism of the track is given in Results.

ID	Type of material	Length	Service required
2603	SLT-6	162 meter	cleaning (34 min)
2401	SLT-4	70 meter	none
2402	SLT-4	70 meter	Cleaning (34 min)
2601	SLT-6	101 meter	none
2602	SLT-6	101 meter	Cleaning (34 min)

Table 4: The available incoming and required departing trains for this scenario.

Two representational solutions are illustrated below as optional solutions for the scenario and location introduced. The solutions are illustrated using the STAG representation, initially introduced in Train Unit Shunting Problem. For sake of clarity, the sequence of movements and service tasks is omitted. This implies that the indications between movements and service tasks are absent, which is not essential to get insight into the recombination technique. Furthermore, an incoming train access the shunting yard at 'Sein 1', and accordingly moves to track 'SpoorNep 45', visualized in figure 13b. This movement is included in the arrival activity, likewise for the departure activity.

Graphs 7 and 8 are two plan graphs of solutions for this scenario. The conflict-based crossover established differentiation of these solutions according to the criteria in Conflict-based selection of Trains. The parallel comparison of the conflicts, neighbours, and overlap of each train unit resulted in the following distribution. The solution of train units 2603 and 2601, 2402 performed superior in parent A, whereas train units 2401, and 2601 in parent B.

First of all, the conflict-based crossover, described in Train Unit Shunting Problem considers the criteria per train unit. However, to perform valid splits after arrival, the train units are selected as entire trains of each solution. In this instance, if train unit 2402 is selected from parent A (figure 7) and 2601 from parent B (figure 8). The train arrives at track 45 and afterwards, 2402 is assigned to track 6 and 2601 to track 1112. At the gateway track 45, it is not allowed to park, and thus separate or combine train units. In this case, there is no option to split the train units before they advance to their next track. To ensure a valid way for trains to arrive and separate for different tracks, an entire train is selected.

The elements highlighted by the azure blue separations are the sequence of movements and tracks selected from parent A, likewise the cobalt blue part for parent B. The incoming trains eventually split into distinct trains to compose the departing trains. The departing composition determines the splits and combines in the remainder of the plan graph. 5 provides an overview of possible configurations of the five incoming train units assigned to the three outgoing trains. Many train unit compositions are available, however, assigning a train unit to a departing train limits the options

tremendously. Finding two solutions for which a combination is possible is very complex. If one assigns two train units to the SLT-6 SLT-4 departing train, there is only a single configuration left to assign the remaining train units. The number of configurations expands enormously by increasing the number of train units at a shunting yard. However, combining the matching of solutions is complex and only possible for a limited number of combinations of solutions.

Departing trains	Composition 1	Composition 2	Composition 3	Composition 4
SLT-6, SLT-4	2603, 2401	2603, 2401	2603, 2402	2603, 2402
SLT-6	2602	2601	2602	2601
SLT-6, SLT-4	2601, 2402	2602, 2401	2601, 2401	2602, 2401

Table 5: The alternative matching configurations

To address the problem of combining different matching, the matching of parent A is consistently selected. The entirety of the matching of parent A is picked for the offspring which determines the splits and combines in the remainder of the plan graph. The matching of parent A is emphasized by the black highlight and implies which train is assigned to the departing train and the departure movement. The benefit is that the matching is guaranteed to be valid, however, does decrease some diversity within the generation. Selecting a random track would be inefficient and most likely does not result in better-performing solutions.

The selection of the matching from parent A influences the extent of the activities selected by the conflict-based crossover. The succession of tracks that remain from a solution are all movements and activities in advance of the movement towards the last parking activity preceding the departure movement, resulting in the highlighted areas for the parent solutions.

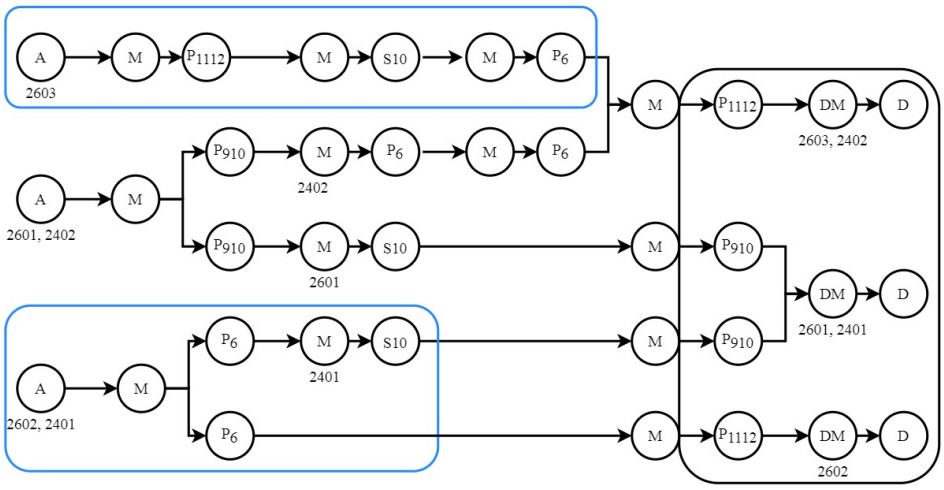


Figure 7: A STAG representation of parent A.

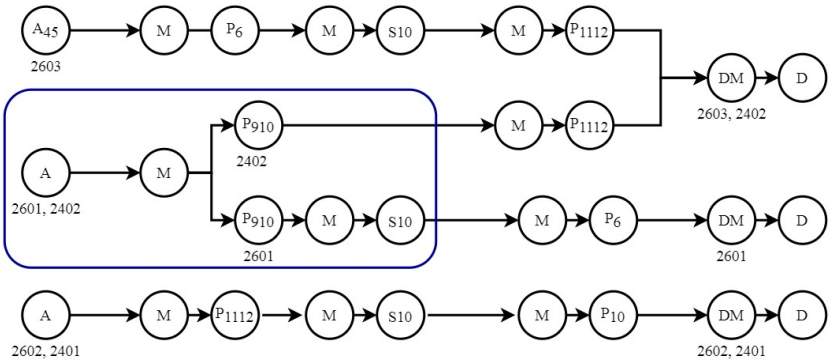


Figure 8: The STAG representation of parent B.

The new solution established by combining parents A and B is illustrated in figure 9. The most substantial part of the recombination is combining the selected parts from the conflict-based crossover to the matching. To ensure a ripple less recombination of train units, the last parking location before a departure movement is stored from solution A. Just for arrival, departing train units are not able to the junction at the departing track. Thus, combining train units is compulsory before this movement, preferably at the last parking location. Train unit 2401 and 2602 are selected from parent B and therefore the track for these train units are adjusted to the last parking track of parent A. Using the parking location of A, guarantees a valid parking location, identical to the parent. Adapting this parking location results in an adjustment to the movement to get there, and the movement to get from the parking location towards the departure movement.

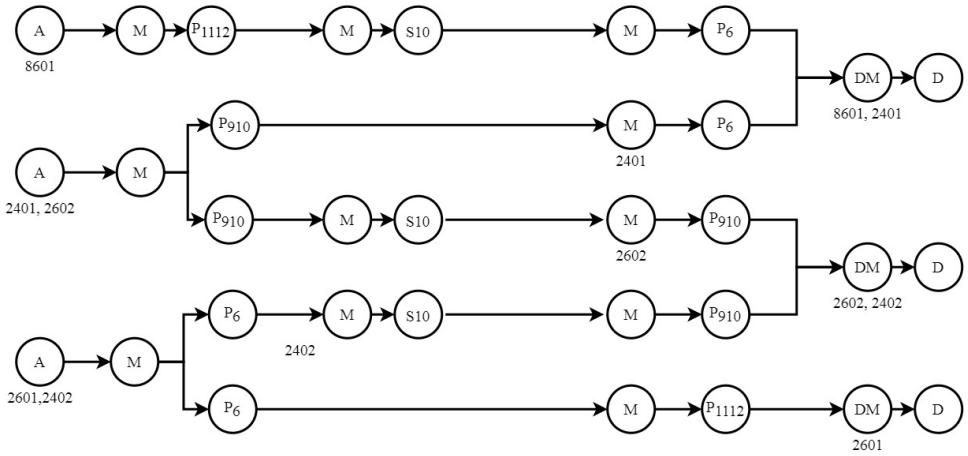


Figure 9: STAG of the resulting solution by mixing solution A and B. Incoming trains 2603 and 2601, 2402 are taken from parent A (azure blue) and train 2401, 2602 from parent B (cobalt blue). The matching is taken from scenario A (black)

Based on the criteria defined in 3.4.3, train units will remain combined, such as train units 2402, 2601 in figure 8. If required, train units separate directly after arrival or just before the departure movement. To maintain this, the algorithm adapts the offspring of two-parent solutions slightly. In the example scenario, train units 2603 2401 combine at track 6, move to track 1112, and eventually depart at track 45. The parking location after the combining track and before the departure movement is superfluous and, therefore not added to the offspring solution.

Adjusting the movements preceding and succeeding the parking location to recombine train units, resulted on occasion

in errors. To find a feasible solution, variation in the solutions is required. To facilitate this, robustness of the algorithm is required. Above is the correction for solutions that include an extra parking location after combining train units. Additionally, if a train does not have a parking location after the service task and before the departure movement, this is added. This is required as a parking location is essential for train units to connect. Without this parking location, the EA was not able to find a correct schedule plan and is not able to continue.

The offspring solution is essentially built up the same way as an initial solution described in Train Unit Shunting Problem. First, the arrival activities of each train are added to the solution graph. Afterward, the matching of parent A is selected and added. Next, the sequence of movements and activities of the corresponding parent for each train unit is added to the solution. Finally, the movement antecedent to the last parking location, the parking location, and the movement succeeding are adjusted to the location of parent A.

5 Results

The overview of a hybrid EA is provided and all elements are suitable to fit the constraints of TUSP. The objective of the experiments is to gain a deeper insight into the performance of the hybrid EA. Furthermore, this section focuses on determining whether the implementation of an EA within the day-to-day planning of NS can improve the current process. Various elements of the EA were tested as described within 3.

The objective of this section is twofold: first of all, the first set of experiments is to obtain the performance of the EA for two types of locations. Scenarios are generated for an ideal carousel and shuffleboard location to evaluate the performance of the EA. Thereafter, the performance of the EA is evaluated and compared to the local search algorithm. This is currently the best-performing algorithm implemented at NS. The performance is important to determine whether implementation of the hybrid EA can help human planners find feasible schedules. This requires the EA to find feasible solutions within a small time frame.

This section provides a setup of the experiments. Next, section 5.2 provides the results of the first experiments in which the performance is measured of the hybrid EA on a carousel and shuffleboard location classified for 6 to 17 train units at the shunting yard. Eventually, the section provides the results of the EA compared to HIP for the different types of locations and number of train units. HIP is used as a reference point, to determine the efficiency of generating feasible shunting yard schedules for NS using an EA.

5.1 Setup of Experiments

First, the configuration of each experiment is discussed. The experiments are conducted to test the performance in different circumstances. The exact methods are given for reproducibility while generating the results of the experiments. To achieve the objective of this research, two experiments are executed.

5.1.1 System

The experiments are run on a virtual machine running on windows server 2019 (64x) The windows server has a 4 core and 32 GB flavor. All experiments are run on machines with the same performance to enable the comparison of results. This is essential for unbiased experiments as the termination criteria are based on time. The software used to run is C# 9.0, supported on .NET 5.0 target framework.

5.1.2 Type of Locations

To determine the underlying reason for the performance for different types of locations, the EA is thoroughly tested on two locations. Most of the real-world locations include a combination of LIFO and free tracks. However, to purely focus on the difference in location, an 'ideal' artificial location is produced. These locations exclusively exist of LIFO-tracks or free tracks. The 'ideal' artificial location ensures a neutral reference point for the hybrid EA. The carousel location is presented in figure 13a and shuffleboard location in figure 13b.

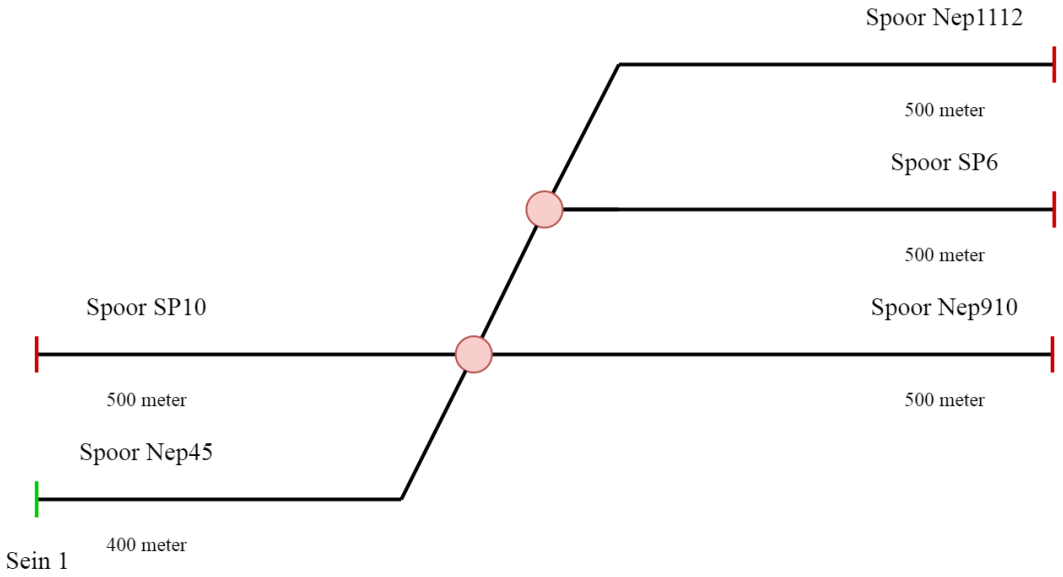


Figure 10: Layout of an ideal four track shuffleboard location.

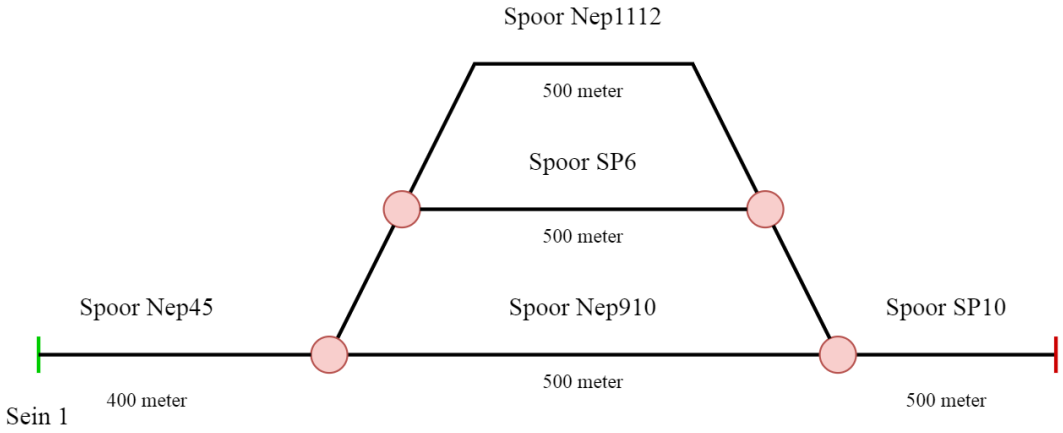


Figure 11: Layout of an ideal four track carousel location.

5.1.3 Problem Instances

The available material for each scenario is provided in table 6. Service tasks are included, whether the material requires service is defined in each scenario. The length of the service tasks is dependent on the material type. Furthermore, the scenarios include combined shunt trains to evaluate the performance of splits and combines in the shunting yard. Next to this, the scenarios include service tasks and the matching is predetermined.

Material	Length
SLT - 4	70 meter
SLT - 6	101 meter
VIRM - 4	109 meter
VIRM - 6	162 meter

Table 6: The available material for the scenario used in the experiments.

The number of tracks for the ideal location is fixed on four tracks, however, the number of incoming and outgoing train units is increasing. The occupation rate for the location heavily influences the complexity of the solution algorithm. Increasing the occupation of the tracks enlarges the probability of conflicts in the shunting yard schedule. The hybrid EA is tested for scenarios including 6 up to 17 train units. Six train units are considered an easy scenario for these locations whereas 17 is considered extremely complex. Scenarios including one to five train units are not considered, the scenarios are very easy and it is most likely that the initial solution is feasible.

5.1.4 Performance Metrics

To interpret the results of experiments, various performance metrics are described to gain insight. The performance metrics are classified for each number of train units. The amount of train units at a shunting yard heavily determines the complexity.

1. *Feasibility:* The most substantial performance metric is the feasibility rate of the solution approach. Finding feasible solutions is the objective of the solution approach.
2. *Number of Generations* The number of generations computed indicates the computational power required to converge to a feasible solution.
3. *Total Evaluations* In addition to the number of generations, the total evaluations are determined. This indicates the impact of the local search method in the EA.
4. *Diversity* This metric represents the diversity within a generation. This is an indication of the size of the search space.

5.1.5 Termination Criteria

Within a constraint feasibility problem, the algorithm is terminated once a feasible solution is found. This is the case for this algorithm, however, to avoid running the algorithm indefinitely, additional termination criteria are used. Possibilities for termination criteria is an iteration limit, average time, number of crossovers, or if the improvement per generation drops below a certain percentage. The ultimate objective is for the solution approach to help human planners provide feasible schedules. To obtain these schedules, a maximum amount of time is preferred, accordingly, NS mainly tests the performance of the local search for a specified time limit. This is the reason that the EA and local search algorithm include a time limit. The EA might take more iterations to converge solutions to a feasible area. Various experiments to evaluate the performance of HIP are conducted for similar scenarios with a time limit between 120 and 300 seconds. The experiments for the scenarios in this research are restricted with a time limit of 240 seconds.

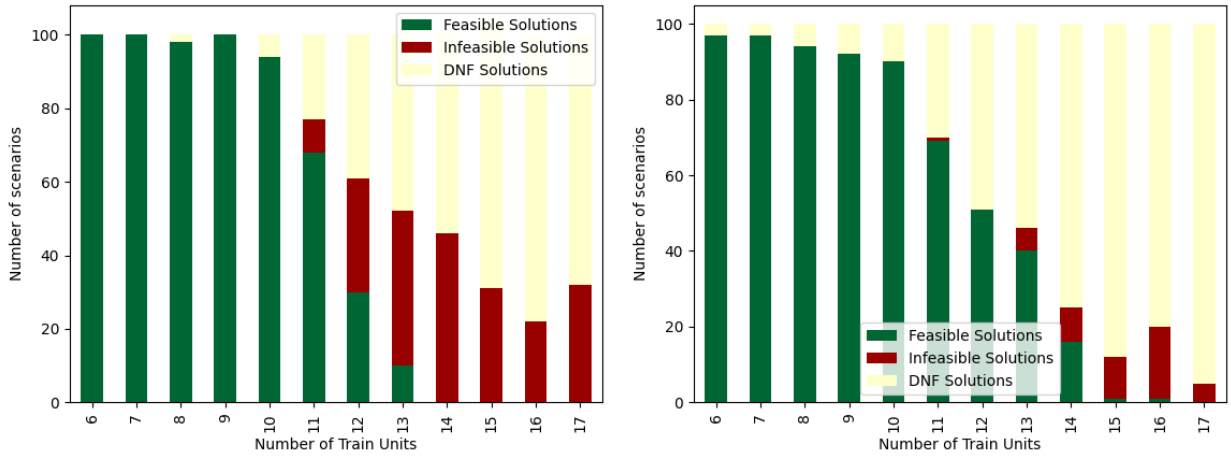
5.2 Comparison of EA performance on Shuffleboard and Carousel Locations

The first experiment evaluates the performance of the EA on the ideal shuffleboard and carousel locations. The exact configuration for the experiments is given in table 7.

Population	8	**Train Count**	6 to 17 train units
Problem instances	1200	**Evaluation limit**	None
Time limit (sec)	240	**Locations**	ideal carousel & shuffleboard

Table 7: Setup of the first experiments

The performance for the carousel and shuffleboard location is provided in figure 12. For each of the two locations, 100 scenarios are performed for each number of train units. The feasible solutions are represented in green, infeasible solutions in red and scenarios that were not able to be fully executed in yellow. The number of runs that were not able to complete especially increased for scenarios with a higher occupation rate. These scenarios generated an incorrect shunting yard schedule which caused it to terminate in an early stadium, the underlying cause for this is explained in the Conclusion.



(a) Carousel location

(b) Shuffleboard location

Figure 12: The performance for the hybrid EA on the carousel and shuffleboard location.

The results suggest that the EA performs better for a shuffleboard location than the carousel location. The EA maintains a high feasibility rate for the shuffleboard up until 11 train units, afterwards, the performance declines. However, even for highly occupied shunting yards with 14 train units it manages to produce 14 feasible schedules. The EA results in 1 feasible schedule for extremely hard instances including 15 and 16 train units. On the other hand, the performance for the carousel location decreases significantly for 12 train units and higher. It is incapable of generating a feasible solution for instances of 14 train units and more.

Additionally, the number of infeasible solutions is higher for the instances performed on carousel locations than in shuffleboard locations. For both the locations, the ones which were not able to complete dominate the higher occupied instances. However, a larger segment of the solutions resulted in an infeasible solution after 240 seconds for the carousel location than for the shuffleboard location.

Table 8 provides an overview of the average number of generations and total evaluations for each number of train units, sorted for the different types of locations. The number of observations declines for higher number of train units, since the number of scenarios declines.

The number of generations and evaluations follow an expected trend for the feasible solutions for both locations. A more complex service site schedule, specifically a higher number of train units to distribute, requires more computational capacity. However, the number of generations required to find a feasible solution is much lower for a

shuffleboard location.

Furthermore, the number of generations for infeasible solutions is according to expectations. The solution algorithm continues computing until it reaches the time limit of 240, to advance towards a feasible solution. The difference of generations completed for carousel and shuffleboard locations differs substantially. A considerable more amount of generations are accomplished for the shuffleboard location in the same amount of time.

	Carousel Location				Shuffleboard Location			
Train Units	number of generations		Total Evaluations		number of generations		Total Evaluations	
	feasible	infeasible	feasible	infeasible	feasible	infeasible	feasible	infeasible
6	1.98	N/A	5,465.59	N/A	1.70	N/A	2,368.36	N/A
7	2.00	N/A	11,137.90	N/A	1.54	N/A	3,775.18	N/A
8	2.26	N/A	33,579.15	N/A	1.60	N/A	5,227.39	N/A
9	2.41	N/A	28,921.93	N/A	1.66	N/A	7,904.23	N/A
10	2.56	N/A	35,824.88	N/A	1.93	N/A	8,306.62	N/A
11	4.70	21.5	80,207.25	296,431.88	2.03	46	13,283.75	130,941.30
12	9.65	25.48	115,042.72	295,089.23	2.92	N/A	17,714.06	N/A
13	9.40	23.44	163,664.40	211,774.74	3.11	44.69	26,613.97	38,9406
14	N/A	15.71	N/A	51,661.00	7.58	34.86	116,633.33	18,071.42
15	N/A	17.67	N/A	39,000.00	12.00	35.00	10,000.00	19,975.00
16	N/A	18.32	N/A	40,909.00	14.00	31.07	10,900.00	179,286
17	N/A	17.59	N/A	98,427.59	N/A	28.00	N/A	164,800

Table 8: The average number of generations and total evaluations per number of train units, for the carousel and shuffleboard location.

The diversity indicates the dissimilarity within the solutions of a generation, in more detail clarified in 3. The diversity of a generation and the mean diversity of an individual might give more insight into the exploration of the search space. Higher diversity is expected for a carousel type of location, due to the higher flexibility at the service site. Before analyzing, some data cleaning was required as some diversity metrics resulted in extreme values. The diversity and diversity per individual solutions exceeding 100.000 were removed from the data set since this is not a realistic quantity.

To gain a deeper understanding of the difference in performance for the different types of locations, other performance metrics are evaluated. The diversity of a generation and the mean diversity of an individual might give more insight into the exploration of the search space. Additionally, the conflict cost before and after the crossover can indicate the impact of the recombination technique. This can give a better understanding of the lower feasibility rate for carousel locations. Furthermore, the number of generations and total iterations might give a better understanding of the in-depth exploration. It is expected that scenarios on the shuffleboard locations realize more in-depth exploration. Table 9 provides an overview of the average diversity and diversity per individual per location, classified by the number of train units. The increased complexity result in an expansion of solution diversification. However, The feasible solutions indicate a plateau of diversity within the solutions. This seems reasonable, as the four-track location limits the search space.

	Carousel Location				Shuffleboard Location			
Train Units	Diversity		Diversity/Individual		Diversity		Diversity/Individual	
	feasible	infeasible	feasible	infeasible	feasible	infeasible	feasible	infeasible
6	256.05	N/A	31.54	N/A	189.89	N/A	22.90	N/A
7	342.65	N/A	42.41	N/A	212.57	N/A	26.14	N/A
8	480.13	N/A	59.59	N/A	287.23	N/A	29.36	N/A
9	485.02	N/A	60.15	N/A	298.77	N/A	37.15	N/A
10	500.44	N/A	63.15	N/A	313.43	N/A	43.37	N/A
11	553.28	436.74	74.93	81.03	308.11	N/A	41.39	24.97
12	487.74	336.17	117.75	64.22	256.48	N/A	62.12	N/A
13	420.57	354.94	57.65	87.65	288.36	126.38	44.69	90.62
14	N/A	324.82	N/A	112.56	158.54	79.45	122.88	164.13
15	N/A	77.97	N/A	489.29	31.17	104.091	33.75	147.94
16	N/A	243.80	N/A	259.33	N/A	98.73	N/A	135.99
17	N/A	102.61	N/A	145.20	N/A	95.83	N/A	114.67

Table 9: The average diversity and diversity per individual for the carousel and shuffleboard location per number of train units.

5.3 Comparison to Local Search Algorithm

The previous experiments compared the performance of an EA on the carousel and shuffleboard locations. However, the performance of the EA compared to HIP is more important to determine potential adoption for NS as this shows whether EA is an improvement when compared to HIP. The specific configuration for the experiments is provided in 10.

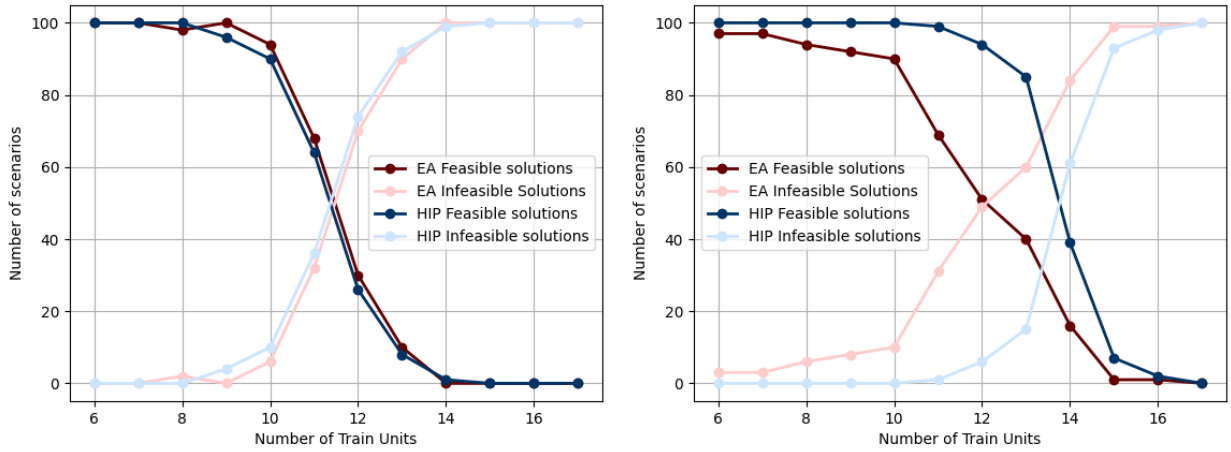
Problem instances	1200	**Train Count**	6 to 17 train units
Iteration limit	None	**Methods**	EA & HIP
Time limit (sec)	240	**Locations**	ideal carousel & shuffleboard

Table 10: The setup configuration for the first experiment/

The results for the experiments are presented in figure 13. The hybrid EA accomplishes a slightly higher feasibility rate for most of the number of train units for a carousel location. The EA finds a higher number of feasible solutions for 9 to 13 train units. On the other side, the EA finds considerably fewer solutions for shuffleboard scenarios than HIP.

The experiments which were not able to complete are considered infeasible. For the reason that HIP exclusively results in feasible or infeasible. However, the unfinished results are considered potentially feasible solutions. The incomplete scenarios can be considered as potentially feasible solutions. However, it is unrealistic to consider the unfinished scenarios for a high number of train units as feasible. Therefore, the potential solutions are calculated using the ratio of feasible to all scenarios for each number of train units, according to the next formula:

$$\frac{Numberoffeasibleexperiments}{Numberoftotalexperiments} * Numberofunfinishedexperiments$$



(a) Carousel location

(b) Shuffleboard location

Figure 13: The performance for the hybrid EA compared to HIP on the carousel and shuffleboard location, including feasible solutions (green), infeasible solutions (red) and solutions that did not finish (DNF) in yellow.

	Carousel location					Shuffleboard location				
Train Unit	HIP	EA	EA Potential	% EA	% potential EA	HIP	EA	EA Potential	% EA	% potential EA
6	100	100	100	0.0 %	0.0 %	100	97	99.91	-3.0 %	-0.1 %
7	100	100	100	0.0 %	0.0 %	100	97	100	-3.0 %	0.0 %
8	100	98	99.96	-2.0 %	0.0 %	96	94	99.64	-6.0 %	-0.4 %
9	96	100	100	4.2 %	4.0 %	100	92	99.36	-8.0 %	-0.6 %
10	90	94	99.64	4.4 %	9.7 %	100	90	99	-10.0 %	-1.0 %
11	64	68	89.08	6.3 %	28.2 %	99	69	91.08	-30.3 %	-8.0 %
12	26	30	41.7	15.4 %	37.6 %	94	51	75.48	-45.7 %	-19.7 %
13	8	10	14.8	25.0 %	45.9 % %	85	40	66.4	-52.9 %	-21.9 %
14	1	0	0	- 100.0 %	NA	39	16	28	-59.0 %	-28.2 %
15	0	0	0	NA	NA	7	1	1.88	-85.7 %	-73.1 %
16	0	0	0	NA	NA	2	1	1.8	-50.0 %	-10.0 %
17	0	0	0	NA	NA	0	0	0	NA	NA

Table 11: Number of feasible solutions HIP vs EA

Table 11 provides an overview of the feasible solutions of the methods and the type of locations. The EA potential covers the incomplete solutions present in the data. The feasibility trend for the particular number of train units is continued for the incomplete scenarios to result in the potential of the EA. The percentage difference between the EA and HIP is provided for which the performance of the EA is slightly higher. The percentage potential of the EA increases from 9 train units onwards to 45 %. The HIP has been developed for TUSP from the first introduction onwards. The slightly higher performance and potential for the current EA are promising on carousel locations. The EA could be adjusted more to TUSP to improve the solutions.

To formally compare the results of HIP and EA, a test of significance is conducted. The results of HIP and EA per train unit are assessed to determine a significant difference. The data used are the 100 scenarios per train unit per method with the result feasible (1) or infeasible (0). First, statistics tests require multiple assumptions to be satisfied to have a greater probability of detecting true differences. The data of the research is evaluated for normality, using a Jarque-Bera test. This determines whether the data follows a normal distribution by evaluating the skewness and kurtosis. The Jarque-Bera test on the carousel location for 12 train units for HIP(M = 0.64; SD = 0.048) and EA (M=0.68; SD =

0.047) was significant (t(2) = 17.65; textitp = 0.00015). With a two tailed test, the result is considered significant if $p< 0.05$. The null hypothesis is rejected and the data is not normally distributedT. K. Kim and Park (2019). This indicates a non-parametric test as this requirement is not satisfied. However, the violation of the normality assumption should not cause major problems for large sample sizes (n > 40) Ghasemi and Zahediasl (2012). Thus, parametric procedures can be used even if the data does not follow a normal distribution. Two Null hypotheses are evaluated to determine the significant difference between the performance of the two types of locations, which are:

The mean of feasible solutions of HIP is the same as the EA for each number of train units at the carousel location.

The results of the t-test for the carousel location are provided below. The number units for which a significance test is presented are 9 to 12 train units for the carousel location. For the other number of train units, the t-test does not result in a significant difference. The result is considered significant if $p< 0.05$.

1. The mean feasible solutions generated by HIP (M=0.96; SD = 0.039) was lower than the performance of the EA (M=1.00; SD = 0.0), and significant ($t(198) = 1.97$; $p = 0.044$) for 9 train units.
2. The mean feasible solutions generated by HIP (M=0.90; SD = 0.091) was lower than the performance of the EA (M=0.94; SD = 0.057), this was not significant ($t(198) = 1.97$; $p = 0.30$) for 10 train units.
3. The mean feasible solutions generated by HIP (M=0.64; SD = 0.23) was lower than the performance of the EA (M=0.68; SD = 0.22), this was not significant ($t(198) = 1.97$; $p = 0.55$) for 11 train units.
4. The mean feasible solutions generated by HIP (M=0.26; SD = 0.19) was lower than the performance of the EA (M=0.30; SD = 0.21), this was not significant ($t(198) = 1.97$; $p = 0.53$) for 12 train units.

The mean of feasible solutions of HIP is the same as the EA for each number of train units at the shuffleboard location.

1. The difference in feasible solutions for HIP(M = 1.0; SD = 0.0) and EA (M = 0.97; SD = 0.03) was not significantly higher ($t(198) = 1.97$; $p = 0.08$) for 6 and 7 train units.
2. The difference in feasible solutions for HIP(M = 1.0; SD = 0.0) and EA (M = 0.94; SD = 0.057) was significantly higher ($t(198) = 1.97$; $p = 0.0$) for 8 train units.
3. The difference in feasible solutions for HIP(M = 0.85; SD = 0.13) and EA (M = 0.40; SD = 0.24) was significantly higher($t(198) = 1.97$; $p < 0.001$) for 13 train units.
4. The difference in feasible solutions for HIP(M = 0.07; SD = 0.066) and EA (M = 0.01; SD = 0.01) was significantly higher ($t(198) = 1.97$; $p = 0.03$) for 14 train units.

Figure 12 indicates considerably lower performance for the EA. The scientific test is conducted for 6 up to 14 train units for the experiments. The scientific test resulted in a significant difference in feasible solutions for HIP compared to the EA on the shuffleboard location for all number of train units from 7 to 14. The summation above provides the significance tests for the number of train units on edge values. The performance difference for the other number of train units between HIP and the EA was not significant.

5.4 Discussion

Section 5 provides an overview of the results of the two experiments conducted in this research. The first experiment demonstrates that the performance of the EA on a shuffleboard location higher than that on the carousel location. Due to the flexibility of carousel locations and thus a broader search space, a higher performance of the EA was expected. One would intuitively assume that more feasible solutions are generated with more options to arrive and depart a track. Increasing the routing options for shunt trains expands the possibilities to generate a schedule without conflicts.

The free tracks available at carousel locations extend the possibilities and therefore, the size of the search space. The benefit of a hybrid EA is the ability to explore and cover a large search space. Due to the combination with the local

search, the solutions eventually converge to a more promising area. This characteristic increases the performance expectation for carousel-type locations compared to shuffleboard locations. The recombination procedure is more effective for a larger search space. Notably, the experiments are restricted to a time limit of 240 seconds which might be more restricting to the performance of the carousel location. Particularly, a considerably larger amount of the experiments are considered infeasible of finding a shunting yard schedule in the given time.

To gain a deeper understanding, the number of generations and total evaluations are provided in table 8. The number of generations required for the EA to find feasible solutions is considerably lower for a shuffleboard location than for a carousel location. The EA requires 43.19% fewer generations to find a feasible solution for a carousel location than a shuffleboard location for an instance of 11 train units. The total evaluations for each location indicate a similar tendency. In addition, a considerably higher amount of evaluations are required to find a feasible solution for scenarios at a carousel location. The required higher number of generations does indicate more complexity to converge to a feasible solution. These results do support the corresponding number of feasible schedules for each type of location of figure 12. The larger search space requires an increased exploration for the EA, given the same amount of time, resulting in fewer feasible solutions.

Furthermore, the diversity of solutions at the carousel location is higher than at the shuffleboard location. This is in line with expectations since a carousel location provides a larger search space for the EA to discover. Next to this, this is in line with the required number of generations and evaluations resulting in an inferior amount of feasible solutions. As the search space is larger, the more computational time is required to converge to a feasible service site schedule. Due to the limited computational time, a limited amount of feasible solutions are accomplished.

The EA is cumulative able to find more feasible solutions for the shuffleboard locations compared to the carousel locations. However, bench-marking these results to HIP indicates a superior performance of the EA for the carousel location. The amount of feasible solutions is slightly, but consistently higher for the EA compared to HIP. The cumulative amount of feasible solutions is inferior for carousel-type of locations for both methods in the limited amount of time. This implies that the solution approaches experience more complications to achieve feasible solutions. The potential performance of the EA indicates a considerable improvement in feasible solutions for carousel locations. The larger search space of the carousel location fits the explorative character of an EA.

However, the performance of the EA compared to HIP for shuffleboard locations is considerably lower. The performance of HIP expands proportionally compared to the EA. The difference between HIP compared the EA for 13 train units increases to 52.9 %. The EA finds considerably less feasible solutions than HIP for the increased occupation rate scenarios. Taking the potentially feasible schedules for the EA into account shows a difference. For more simple scenarios it might perform equally well, however for a higher number of train units and therefore more complex scenarios HIP outperforms the EA solution approach for shuffleboard locations.

This research is the first to apply an EA to TUSP for all scenarios including splits and combines. However, Athmer (2021) introduced an EA to solve scenarios without splits and combines for TUSP. This research evaluates the performance of HIP and an EA for the real-world locations 'Grote Binckhorst' and 'Kleine Binckhorst'. These locations can be considered shuffleboard and carousel locations. A comparison of shuffleboard to carousel performance is not possible as the experiments differ concerning location size, number of tracks, and number of train units. Furthermore, these experiments include an evaluation limit of 1.600.000 and no time limit which differs from this study. Nevertheless, the EA can solve 99% of the scenarios compared to 62% for HIP on the carousel location. Whereas, the performance for the shuffleboard location results in 10% by the EA in contrast to the 100 % of HIP. The tendency of the performance of the EA is largely similar to the addition of splits and combines.

In conclusion, the EA can find a higher number of total feasible solutions for the shuffleboard locations. Experiments demonstrate that the EA requires fewer generations and evaluations to successfully find solutions. Next to this, the diversity within generations indicates a smaller search space to explore which results in more feasible solutions for the limited amount of time in the experiments. However, comparing the performance of the EA to HIP, it significantly performs inferior. The local search can find more feasible solutions than HIP. Finally, the EA is consistently able to find more feasible solutions compared to HIP for carousel locations.

6 Conclusion

First of all, a literature framework is presented of current research about TUSP. This indicates that exact solutions are not suitable, concerning computational time. At this point HIP, a local search method, is the best performing algorithm to solve real-world scenarios. NS is integrating the automated planner system to find feasible solutions to optimize the use of their service sites. The hybrid integrated planner is the first algorithm capable of solving TUSP including all matching, parking, routing and servicing constraints for real-world instances.

This study aims to increase the performance of finding feasible solutions using a hybrid evolutionary algorithm to TUSP. The Literature Review presents an overview of EA applications to complex feasibility problems. However, literature did not present a recombination technique able to integrate splits and combines for TUSP, answering research question 1. Literature indicates that EAs are able to satisfy constraints using direct and indirect constraints. Furthermore, to generate solutions that converge towards a promising area of the search space, a problem-dependent recombination technique is essential.

As discussed in the Introduction, this research set out a goal to determine whether it is possible to include splits and combines in TUSP. To address this issue, this research presents a new recombination technique to find feasible shunting plans including splits and combines using an EA. This technique allows one to recombine the best-performing elements of two parents to construct a new offspring solution. This is realized by connecting the shunt trains from one parent to the matching of the other parent. The procedure modifies the movements preceding and succeeding of the parking location, allowing shunt trains to combine, thereby confirming that EA is capable of finding TUSP solutions including combines and splits.

The performance of the hybrid EA is evaluated on two types of locations, namely an ideal four-track carousel and shuffleboard locations. The EA is able to find substantially more feasible service site schedules for shuffleboard locations than carousel location. The number of generations and evaluations required to find feasible solutions is considerably lower for shuffleboard locations. Next to that, the diversity within a generation on the carousel location is higher than that of a shuffleboard location. The increased flexibility due to free-tracks for carousel locations results in a larger search space, which requires the EA more iterations to converge towards a feasible solution. Thus, the EA is able to find more feasible solutions for the shuffleboard location in the limited amount of time.

Furthermore, this study aims to determine in which circumstances the performance of the EA is superior compared to that of HIP. Both solution approaches were tested on a four-track ideal carousel and shuffleboard location, classified for 6 to 17 train units. Remarkably, the EA was consistently able to find more feasible shunting schedules for scenarios at the carousel location. Moreover, it performed significantly higher than the currently-in-use solution algorithm for 12 train units for this location, despite the years of research behind HIP. On the other hand, HIP performed significantly better on the shuffleboard location than the EA solution approach. The performance of the EA can improve to the potential after further investigation into the method.

Overall, the configuration of the EA demonstrates a promising solution approach for TUSP including the option to split and combine. The performance for a shuffleboard location was inferior compared to HIP, however it surpassed for a carousel location. The extended flexibility of carousel location is in harmony with the characteristic of a hybrid EA to explore a large search space. It demonstrates a competitive performance for carousel locations in a limited amount of time compared to the state-of-the-art method. The hybrid EA establishes a promising performance and might be beneficial with further research to eventually achieve a higher performance of feasible shunt schedules for NS.

6.1 Limitations

The different elements of an EA allow flexibility to fit the corresponding problem. This allows the EA to handle the complex interactions between several NP-hard sub-problems. However, the currently developed EA includes some shortcomings which are described in this section.

Firstly, the EA is not able to complete all experiments and consistently achieve results. This is the first crucial shortcoming of the current EA. Adjusting the movements preceding and succeeding plus the parking location does not

guarantee a correct schedule graph. To increase robustness, alternations of the standard procedure are covered. A parking location can be absent for initial solutions that are generated as described in section 3.4.3. To address this problem, a parking location is inserted to facilitate the recombination procedure. Meanwhile, the local search algorithm adjusts the precedence order schedule to improve the solution. The current EA does not achieve the level of robustness to manipulate these adjustments to create a polished recombination procedure. This is especially reflected in the complex scenarios for which the EA requires numerous generations.

The train units resulting in the least amount of conflicts are chosen from each parent. Consequentially, these shunt trains are consistently connected to the matching of parent A, regardless of the number of shunt trains selected from either parent. This can eventually result in homogeneity in the matching of a generation, especially over multiple generations. However, the combination of two matching schemes is very complex, due to the interaction. Selecting an arrival train for a designated departure train severely limits the options.

The current EA selects the combination of parents using the diversity metric. This procedure transforms a STAG representation into a string representation to quantify the difference between solutions. It determines the minimum number of alterations required to convert one solution into another. The matchmaking of parents is based upon the two solutions with the highest diversity. This aims to combine two diverse solutions to create a solution in a new part of the search space. This can result in multiple pairs with the same solution, despite its quality. This eventually reinforces the undesired effect of selecting the matching of parent A described above. Additionally,

This research demonstrates experiments using the termination criteria of 240 seconds. The additional procedures within an EA requires more computation. The higher diversity for carousel-type location indicate a larger search space. The EA requires more computation time to explore the search space to eventually converge to a feasible solution. Within the given time limit, the EA was not able to achieve nearly as much generations for the carousel location as for the shuffleboard location. The performance of the EA could benefit a higher computation time, especially for more complex instances.

6.2 Future Research

The most essential improvement for this EA is improving the robustness. Due to errors within the EA, during the running of the algorithm, it encountered errors that caused infeasible solutions. This especially occurred in scenarios with a high amount of train units and complexity. Enhancing the robustness allows the EA to explore a larger part of the search space, since new solutions can be covered. Improving the robustness will determine the true capabilities of an EA and whether this is an improvement to HIP.

The moment to execute the procedure of splitting and combining is constant within the EA. The splitting procedure is performed as soon as possible after arrival at the shunting yard. The combination procedure is performed at the parking location just before departure. If shunt trains assigned to the same departing train are at the same track, it would be convenient to combine them. This combination is possible if the shunt trains are in identical order, do not exceed track limitations, and, do not surpass the departure time. If two shunt trains are at the same track, these criteria need to be evaluated before combination. Furthermore, this influences the succeeding movements of these shunt trains directly. Including the option to split and combine trains throughout the process, instead of only at arrival or before departure, improves the flexibility of the EA but also significantly increases the complexity. For future research it might be interesting to see whether this also significantly improves feasibility of outcomes.

The selection of elements of each parent is based on the conflict-based crossover. The resulting conflicts of each shunt train are calculated and the best-performing shunt train of a parent is selected. At this point, the conflicts include the indirect constraints of the objective value. These are the utmost important features to optimize as these result in a feasible solution. If these conflicts result in a draw, the neighbour count of each unit is considered. However, combination procedures introduce new conflicts. These conflicts can be avoided if the following are included in the neighbour count. Namely, two trains that are assigned to the same departing train can be either on a different track or on the same track however another train unit in between or at the same track but in the wrong order.

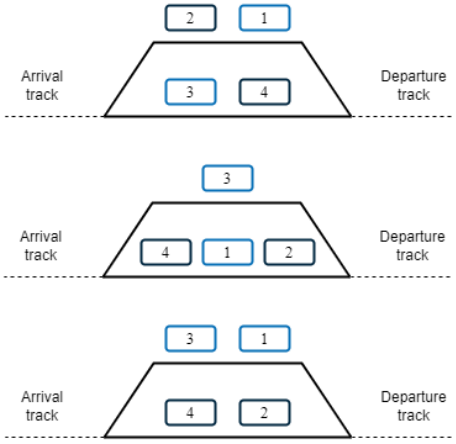


Figure 14: The optional complications for a combination procedure for train units 2 and 4, namely a wrong track, the right track with a train unit in between and a wrong train unit order.

Once the algorithm is further optimized to fit TUSP including splits and combines, it is interesting to evaluate the performance for various shunting yards. This research focuses on the difference of two ideal shunting yards. However, NS deals with combinations of carousel and shuffleboard locations and due to space restrictions does not only have ideal shunting yards. Further research could look at the performance of EA compared to HIP for combined type or non-ideal shunting yards. Nonetheless, the results of the experiments demonstrate a considerable difference for the type of shunting yards of the EA. The EA is able to outperform HIP on a carousel location, however is inferior for a shuffleboard location. The performance gap of the EA compared to HIP, indicate that the EA is specifically appropriate for carousel locations. To address this issue, NS could introduce an algorithm selection feature. To maximize the feasible shunting schedules, the EA is applied to carousel locations and HIP for shuffleboard locations. If a feature is introduced that classifies shunting yards in type of locations, the best performing solution algorithm is applied.

List of Figures

List of Tables

References

Akçay, Y., Li, H., & Xu, S. H. (2004, 3). The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, *155*(1), 1–21. Retrieved from https://ideas.repec.org/a/eee/ejores/v155y2004i1p1-21.html doi: 10.1007/S10479-006-0150-4

Ali, I. M., Essam, D., & Kasmarik, K. (2020, 7). Differential Evolution Algorithm for Multiple Inter-dependent Components Traveling Thief Problem. *2020 IEEE Congress on Evolutionary Computation, CEC 2020 - Conference Proceedings*. doi: 10.1109/CEC48606.2020.9185692

Amaya, J. E., Porras, C. C., & Fernández Leiva, A. J. (2015, 1). Memetic and hybrid evolutionary algorithms. In *Springer handbook of computational intelligence* (pp. 1047–1060). Springer Berlin Heidelberg. doi: 10.1007/978-3-662-43505-2{_}52

Andreica, A., Chira, C., Andreica, A., & Chira, C. (2015). Best-order crossover for permutation-based evolutionary algorithms. *Appl Intell*, *42*, 751–776. doi: 10.1007/s10489-014-0623-0

Athmer, C. (2021, 7). *An Evolutionary Algorithm for the Train Unit Shunting and Servicing Problem.* Delft. Retrieved from https://repository.tudelft.nl/islandora/object/uuid%3A0a65376e-bbb1-440b-95a2-586a369d9e98

Bagchi, T. (1999). The Nondominated Sorting Genetic Algorithm: NSGA. In *Multiobjective scheduling by genetic algorithms.* Retrieved from https://link.springer.com/content/pdf/10.1007%2F978-1-4615-5237-6_8.pdf

Bain, S., Thornton, J., & Sattar, A. (2004). Evolving algorithms for constraint satisfaction. In *Proceedings of the 2004 congress on evolutionary computation, cec2004* (Vol. 1, pp. 265–272). doi: 10.1109/cec.2004.1330866

Bärecke, T., & Detyniecki, M. (2007). Memetic algorithms for inexact graph matching. *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, 4238–4245. doi: 10.1109/CEC.2007.4425024

Bean, J. C. (1994, 5). Genetic Algorithms and Random Keys for Sequencing and Optimization. *https://doi.org/10.1287/ijoc.6.2.154*, *6*(2), 154–160. Retrieved from https://pubsonline.informs.org/doi/abs/10.1287/ijoc.6.2.154 doi: 10.1287/IJOC.6.2.154

Bourret, P., & Gaspin, C. (2003, 1). A neural based approach of constraints satisfaction problem. , 588–593. doi: 10.1109/IJCNN.1992.227255

Broek, R. v. d. (2016). *Train Shunting and Service Scheduling: an integrated local search approach.* Utrecht. Retrieved from http://dspace.library.uu.nl/handle/1874/338269http://localhost/handle/1874/338269

Bull, L. (1997). Model-based evolutionary computing: A neural network and genetic algorithm architecture. In *Proceedings of the ieee conference on evolutionary computation, icec* (pp. 611–616). doi: 10.1109/icec.1997.592384

Chen, Y.-p., & Lim, M.-h. (2008). Linkage in Evolutionary Computation. , *157*. Retrieved from http://link.springer.com/10.1007/978-3-540-85068-7 doi: 10.1007/978-3-540-85068-7

Cicirello, V. A. (2006). Non-Wrapping Order Crossover: An Order Preserving Crossover Operator that Respects Absolute Position. *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*. doi: 10.1145/1143997

Coello, C. A. C. (n.d.). Constraint-Handling Techniques used with Evolutionary Algorithms. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. Retrieved from http://dx.doi.org/10.1145/2908961.2926986 doi: 10.1145/2908961

Davis, L. (1985). *Applying Adaptive Algorithms to Epistatic Domains.* Retrieved from https://www.google.com/search?q=.+Davis%2C+%22Applying+Adaptive+Algorithms+to+Epistatic+Domains%22%2C+Proceedings+of+the+9%27%22+International+Joint+Conference+on+Artijiciul+Intelligence%2C+1985.&rlz=1C1GCEU_nlNL821NL821&oq=.+Davis%2C++%22Applying+Adaptive+Algorithms+to++Epistatic++Domains%22%2C+Proceedings+of+the+9%27%22+International+Joint+Conference+on+Artijiciul+Intelligence%2C+1985.&aqs=chrome..69i57.300j0j7&sourceid=chrome&ie=UTF-8

Deb, K. (2001). Multiobjective Optimization Using Evolutionary Algorithms. . *Jhon Wiley and Sons Ltd*. Retrieved from https://www.researchgate.net/publication/220045365_Multiobjective

_Optimization_Using_Evolutionary_Algorithms_Wiley_New_York

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002, 4). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. doi: 10.1109/4235.996017

de Oliveira da Costa, P. R., Rhuggenaath, J., Zhang, Y., Akcay, A., Lee, W. J., & Kaymak, U. (2019, 7). Data-driven Policy on Feasibility Determination for the Train Shunting Problem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11908 LNAI*, 719–734. Retrieved from https://arxiv.org/abs/1907.04711v1 doi: 10.48550/arxiv.1907.04711

de Oliveira da Costa, P. R., Rhuggenaath, J., Zhang, Y., Akcay, A., Lee, W. J., & Kaymak, U. (2020, 9). Data-Driven Policy on Feasibility Determination for the Train Shunting Problem. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 11908 LNAI, pp. 719–734). Springer, Cham. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-46133-1_43 doi: 10.1007/978-3-030-46133-1{_}43

Dincbas, M., Simonis, H., ECAI, P. V. H., & 1988, u. (n.d.). Solving the Car-Sequencing Problem in Constraint Logic Programming. *researchgate.net*. Retrieved from https://www.researchgate.net/profile/Pascal-Van-Hentenryck/publication/220838036_Solving_the_Car_Sequencing_Problem_in_Constraint_Logic_Programming/links/0f317532829d529643000000/Solving-the-Car-Sequencing-Problem-in-Constraint-Logic-Programming.pdf

Dozier, G., Bowen, J., & Bahler, D. (1995). *Solving Randomly Generated Constraint Satisfaction Problems Using a Micro-Evolutionary Hybrid That Evolves a Population of Hill-Climbers* (Tech. Rep.). doi: 10.1109/ICEC.1995.487454

Eiben, A. E. (2001). Evolutionary Algorithms and Constraint Satisfaction: Definitions, Survey, Methodology, and Research Directions. In (pp. 13–30). doi: 10.1007/978-3-662-04448-3{_}2

Eiben, A. E. A., & Smith, J. E. J. (2003). Introduction to evolutionary computing. , 37–69. Retrieved from https://link.springer.com/content/pdf/10.1007/978-3-662-44874-8.pdfhttp://link.springer.com/10.1007/978-3-662-05094-1_3 doi: 10.1007/978-3-662-05094-1{_}3

Fogel, D. (1992). *Evolving Artificial Intelligence* (Doctoral dissertation, University of California, San Diego). doi: 10.4236/OALIB.1105221

Fogel, L. J. (1964). *On the Organization of Intellect* (Doctoral dissertation, University of California, San Diego). doi: 10.4249/SCHOLARPEDIA.1818

Freling, R., Lentink, R. M., Kroon, L. G., & Huisman, D. (2005). Shunting of passenger train units in a railway station. *Transportation Science*, *39*(2), 261–272. doi: 10.1287/TRSC.1030.0076

Garey, M., & Johnson, D. (1979). Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *W.H. Freedman and Co*. Retrieved from https://search.proquest.com/openview/e4a13c8c3d470245a831eb2f3c3d2487/1?pq-origsite=gscholar&cbl=30748

Ghasemi, A., & Zahediasl, S. (2012, 4). Normality Tests for Statistical Analysis: A Guide for Non-Statisticians. *International Journal of Endocrinology and Metabolism*, *10*(2), 486. Retrieved from /pmc/articles/PMC3693611//pmc/articles/PMC3693611/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC3693611/ doi: 10.5812/IJEM.3505

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Retrieved from http://www2.fiit.stuba.sk/~kvasnicka/Free%20books/Goldberg_Genetic_Algorithms_in_Search.pdf

Gomes, R., Farias, G., & De Magalhães, C. S. (2018). *Parent Selection Strategies in Niching Genetic Algorithms; Parent Selection Strategies in Niching Genetic Algorithms*.

Guan, Q., & Friedrich, G. (1993). *Fuzzy Control over Constraint Satisfaction Problem Solving in Structural Design* (Tech. Rep.). doi: 10.1109/FUZZY.1993.327583

Haahr, J. T., Lusby, R. M., & Wagenaar, J. C. (2015, 10). A Comparison of Optimization Methods for Solving the Depot Matching and Parking Problem. *ERIM report series research in management Erasmus Research Institute of Management*.

Haijema, R., Duin, C. W., & Van Dijk, N. M. (2006, 2). Train Shunting: A Practical Heuristic Inspired by Dynamic

Programming. *Planning in Intelligent Systems: Aspects, Motivations, and Methods*, 437–475. doi: 10.1002/0471781266.CH16

Hajri, S., Liouane, N., Hammadi, S., & Borne, P. (2000, 10). A controlled genetic algorithm by fuzzy logic and belief functions for job-shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *30*(5), 812–818. doi: 10.1109/3477.875454

Hara, A., Kushida, J. I., Tanemura, R., & Takahama, T. (2016, 8). Deterministic crossover based on target semantics in geometric semantic genetic programming. *Proceedings - 2016 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016*, 197–202. doi: 10.1109/IIAI-AAI.2016.220

Haralick, R. M., & Elliott, G. L. (n.d.). Increasing Tree Search Efficiency for Constraint Satisfaction Problems.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. doi: 10.1109/TSSC.1968.300136

Hauptman, A., Elyasaf, A., Sipper, M., & Karmon, A. (2009). GP-Rush: Using Genetic Programming to Evolve Solvers for the Rush Hour Puzzle. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*. Retrieved from http://cs.ulb.ac.be/ doi: 10.1145/1569901

Hinterding, R. (2000). Representation, Mutatin and Crossover Issues in Evolutionary Computation. *School of communications and Informatics*. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=870740

Hoffmeister, F., & Bäck, T. (1990). Genetic Algorithms and Evolution Strategies: Similarities and Differences. In (pp. 455–470).

Holland, J. H. (1992a). *Adaptation in natural and artificial systems* . MIT Press.

Holland, J. H. (1992b). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with ... - John Henry Holland, Professor of Psychology and of Electrical Engineering and Computer Science John H Holland, Senior Lecturer in Human Resource Management Holland - Google Boeken* (First MIT Press edi... ed.). Retrieved from https://books.google.nl/books?hl=nl&lr=&id=wS0LEAAAQBAJ&oi=fnd&pg=PR7&ots=PCRJ4pvbhp&sig=aPn2e-caUYn4VcQE7Kld82vWHrQ&redir_esc=y#v=onepage&q&f=false

Hoos, H. H., & Stützle, T. (2005). TRAVELLING SALESMAN PROBLEMS. *Stochastic Local Search*, 357–416. doi: 10.1016/B978-155860872-6/50025-1

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. *IEEE Conference on Evolutionary Computation*, *1*, 82–87. doi: 10.1109/ICEC.1994.350037

Hu, X. B., & Di Paolo, E. (2008). Genetic algorithms for the airport gate assignment: Linkage, representation and uniform crossover. *Studies in Computational Intelligence*, *157*, 361–387. doi: 10.1007/978-3-540-85068-7{_}15

Im, S. M., & Lee, J. J. (2008). Adaptive crossover, mutation and selection using fuzzy system for genetic algorithms. *Artificial Life and Robotics*, *13*(1), 129–133. doi: 10.1007/s10015-008-0545-1

Jacobsen, P. M., & Pisinger, D. (2011, 6). Train shunting at a workshop area. *Flexible Services and Manufacturing Journal*, *23*(2), 156–180. doi: 10.1007/s10696-011-9096-1

Jalali Varnamkhasti, M., Lee, L. S., Abu Bakar, M. R., & Leong, W. J. (2012). A genetic algorithm with fuzzy crossover operator and probability. *Advances in Operations Research*, *2012*. doi: 10.1155/2012/956498

Jekkers, D. (2009, 4). *Train shunt planning using genetic algorithms.* Rotterdam.

Jeong, S., Hasegawa, S., Shimoyama, K., & Obayashi, S. (2009, 11). Development and investigation of efficient GA/PSO-hybrid algorithm applicable to real-world design optimization. *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, 777–784. Retrieved from https://tohoku.pure.elsevier.com/en/publications/development-and-investigation-of-efficient-gapso-hybrid-algorithm doi: 10.1109/CEC.2009.4983024

Ji, J., Guo, Y., Gong, D., & Shen, X. (2021, 6). Evolutionary multi-task allocation for mobile crowdsensing with limited resource. *Swarm and Evolutionary Computation*, *63*. doi: 10.1016/J.SWEVO.2021.100872

Jindal, A., & Bansal, S. (2019, 11). Effective Methods for Constraint Handling in Quantum Inspired Evolutionary Algorithm for Multi-Dimensional 0-1 Knapsack Problem. *2019 4th International Conference on Information Systems and Computer Networks, ISCON 2019*, 534–537. doi: 10.1109/ISCON47742.2019.9036166

Karakatič, S. (2021, 2). Optimizing nonlinear charging times of electric vehicle routing with genetic algorithm. *Expert Systems with Applications*, *164*, 114039. doi: 10.1016/J.ESWA.2020.114039

Karakatič, S., & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing Journal*, *27*, 519–532. doi: 10.1016/J.ASOC.2014.11.005

Kim, G. H., & Lee, C. S. (1995). Genetic reinforcement learning approach to the machine scheduling problem. *Proceedings - IEEE International Conference on Robotics and Automation*, *1*, 196–201. doi: 10.1109/ROBOT.1995.525285

Kim, T. K., & Park, J. H. (2019, 8). More about the basic assumptions of t-test: normality and sample size. *Korean Journal of Anesthesiology*, *72*(4), 331. Retrieved from /pmc/articles/PMC6676026//pmc/articles/PMC6676026/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC6676026/ doi: 10.4097/KJA.D.18.00292

Kowalczyk, R. (1998). *On solving fuzzy constraint satisfaction problems with genetic algorithms* (Tech. Rep.). doi: 10.1109/ICEC.1998.700147

Kroon, L., Lentink, R., & Schrijver, A. (2006, 12). Shunting of Passenger Train Units: an Integrated Approach. *Transportation Science*, *42*(4), 436–449. Retrieved from https://ideas.repec.org/p/ems/eureri/8292.html

Lee, M. A., & Takagi, H. (1993). Integrating Design Stages of Fuzzy Systems using Genetic Algorithms. *Second IEEE International Conference on Fuzzy Systems*. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=327418

Lek, S., & Park, Y. S. (2008, 1). Artificial Neural Networks. *Encyclopedia of Ecology, Five-Volume Set*, 237–245. doi: 10.1016/B978-008045405-4.00173-7

Lentink, R., Fioole, P.-J., Kroon, L., & van 't Woudt, C. (2006, 1). Applying Operations Research techniques to planning of train shunting. *ERIM Report Series Research in Management*, *96*(1-4), 287–315. Retrieved from https://ideas.repec.org/p/ems/eureri/1100.html doi: 10.1023/A:1018907720194

Li, D., Yang, M., Jin, C. J., Ren, G., Liu, X., & Liu, H. (2021, 4). Multi-Modal Combined Route Choice Modeling in the MaaS Age Considering Generalized Path Overlapping Problem. *IEEE Transactions on Intelligent Transportation Systems*, *22*(4), 2430–2441. doi: 10.1109/TITS.2020.3030707

Lim, K. L., Drage, T., Podolski, R., Meyer-Lee, G., Evans-Thompson, S., Lin, J. Y. T., ... Bräunl, T. (2018, 10). A Modular Software Framework for Autonomous Vehicles. *IEEE Intelligent Vehicles Symposium, Proceedings*, *2018-June*, 1780–1785. doi: 10.1109/IVS.2018.8500474

Majeed, H., & Ryan, C. (2006). Using context-aware crossover to improve the performance of GP. *GECCO 2006 - Genetic and Evolutionary Computation Conference*, *1*, 847–854. doi: 10.1145/1143997.1144146

Majeed, H., & Ryan, C. (2007). A new approach to calculate the best context of a tree and its application in defining a constructive, context aware crossover for GP. *Proceedings of the Frontiers in the Convergence of Bioscience and Information Technologies, FBIT 2007*, 765–768. doi: 10.1109/FBIT.2007.100

McCulloch, W. S., & Pitts, W. (1943, 12). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. doi: 10.1007/BF02478259

Morris, P. (1993). The Breakout Method For Escaping From Local Minima. *AAAI-93 Proceedings*. Retrieved from www.aaai.org

Moscato, P. (n.d.). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts Towards Memetic Algorithms.

Moscato, P., & Cotta, C. (2006, 2). A Gentle Introduction to Memetic Algorithms. *Handbook of Metaheuristics*, 105–144. doi: 10.1007/0-306-48056-5{_}5

Moscato, P., & Cotta, C. (2010). A Modern Introduction to Memetic Algorithms. , 141–183. doi: 10.1007/978-1-4419-1665-5{_}6

Nagavalli, S., Chakraborty, N., & Sycara, K. (2017, 7). Automated sequencing of swarm behaviors for supervisory control of robotic swarms. *Proceedings - IEEE International Conference on Robotics and Automation*, 2674–2681. doi: 10.1109/ICRA.2017.7989312

Norman, M., & Moscato, P. (1991). A competitive and cooperative approach to complex combi-natorial search. *Proceedings of the 20th Informatics and Operations Research Meeting, BuenosAires (20th JAIIO),*, 15–3. Retrieved from https://www.google.com/search?q=M.G.+Norman+and+P.+Moscato.+

A+competitive+and+cooperative+approach+to+complex+combi-natorial+search
.&rlz=1C1GCEU_nlNL821NL821&oq=M.G.+Norman+and+P.+Moscato.+A+competitive+
and+cooperative+approach+to+complex+combi-natorial+search.&aqs=chrome.
.69i57.249j0j7&sourceid=chrome&ie=UTF-8

Ong, Y.-S., Lim, M., & Chen, X. (2010). Memetic Computation -Past, Present & Future. *IEEE Computational Intelligence Magazine*, *5*(2), 24–31. Retrieved from https://www.academia.edu/3369940/Memetic_Computation_Past_Present_and_Future_Research_Frontier_

Papavasileiou, E., Cornelis, J., & Jansen, B. (2020). A Systematic Literature Review of the Successors of "NeuroEvolution of Augmenting Topologies". Retrieved from https://doi.org/10.1162/evco_a_00282 doi: 10.1162/evco{_}a{_}00282

Peer, E., Menkovski, V., Zhang, Y., & Lee, W. J. (2019, 1). Shunting Trains with Deep Reinforcement Learning. *Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018*, 3063–3068. doi: 10.1109/SMC.2018.00520

Peer, E., Zhang, Y., Menkovski, V., Lee, W. J., & Huisman, B. (2018). Shunting trains with deep reinforcement learning. *Physical Education and Sport for Children and Youth with Special Needs Researches – Best Practices – Situation*, 175–176. Retrieved from https://research.tue.nl/en/publications/shunting-trains-with-deep-reinforcement-learning-2 doi: 10.2/JQUERY.MIN.JS

Ponsich, A., Jaimes, A. L., & Coello, C. A. (2013). A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. *IEEE Transactions on Evolutionary Computation*, *17*(3), 321–344. doi: 10.1109/TEVC.2012.2196800

Price, K. V., Storn, R. M., & Lampinen, J. A. (2005). *Differential Evolution a practical approach to global optimization*.

Prins, C. (2004, 10). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, *31*(12), 1985–2002. doi: 10.1016/S0305-0548(03)00158-8

Radcliffe, N. J., & Surry, P. D. (1994a). Fitness Variance of Formae and Performance Prediction. , 51–72.

Radcliffe, N. J., & Surry, P. D. (1994b). Formal Memetic Algorithms. *Evolutionary Computing*, *865 LNCS*, 1–16. doi: 10.1007/3-540-58483-8{_}1

Ray, T., Kang, T., & Chye, S. K. (n.d.). An Evolutionary Algorithm for Constrained Optimization.

Rojas, A., Montero, E., & Riff, M. C. (2017, 7). G-DMP: An algorithm without constraint relaxation to solve the train departure matching problem. *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, 2382–2389. doi: 10.1109/CEC.2017.7969593

Rothlauf, F. (n.d.). Representations for Genetic and Evolutionary Algorithms Second Edition Representations for Genetic and Evolutionary Algorithms.

Sanhueza, C., Mendes, A., Jackson, M., & Clement, R. (2020, 7). An efficient genetic algorithm for the train scheduling problem with fleet management. *2020 IEEE Congress on Evolutionary Computation, CEC 2020 - Conference Proceedings*. doi: 10.1109/CEC48606.2020.9185779

Santucci, V., Baioletti, M., & Milani, A. (2016, 10). Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. *IEEE Transactions on Evolutionary Computation*, *20*(5), 682–694. doi: 10.1109/TEVC.2015.2507785

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of the first international conference on genetic algorithms and their applications, 1985*. Retrieved from https://scholar.google.com/citations?view_op=view_citation&hl=en&user=pRy5WdkAAAAJ&alert_preview_top_rm=2&citation_for_view=pRy5WdkAAAAJ:u5HHmVD_uO8C

Semet, Y., & Schoenauer, M. (2005). An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. *2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings*, *3*, 2752–2759. doi: 10.1109/CEC.2005.1555040

Sharma, S., Blank, J., Deb, K., & Panigrahi, B. K. (2021, 8). Ensembled Crossover based Evolutionary Algorithm for Single and Multi-objective Optimization. , 1439–1446. doi: 10.1109/CEC45853.2021.9504698

Shiroma, P. J., & Niemeyer, G. (1998). Production scheduling in the textile industry: A practical approach using evolutionary algorithms with domain-dependent information. *IECON Proceedings (Industrial Electronics*

Conference), *1*, 269–273. doi: 10.1109/IECON.1998.724134

Singh, H., Gupta, M. M., Meitzler, T., Hou, Z. G., Garg, K. K., Solo, A. M., & Zadeh, L. A. (2013, 6). Real-Life Applications of Fuzzy Logic. *Advances in Fuzzy Systems*, *2013*. Retrieved from https://www.hindawi.com/journals/afs/2013/581879/ doi: 10.1155/2013/581879

Slowik, A., & Kwasnicka, H. (2020, 3). Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications 2020 32:16*, *32*(16), 12363–12379. Retrieved from https://link.springer.com/article/10.1007/s00521-020-04832-8 doi: 10.1007/S00521-020-04832-8

Storn, R., & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341–359. doi: 10.1023/A:1008202821328

Thierens, D. (2010). The linkage tree genetic algorithm. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 6238 LNCS, pp. 264–273). doi: 10.1007/978-3-642-15844-5{_}27

Tsang, E. P. K. (1987). The Consistent Labeling Problem in Temporal Reasoning. In *Aaai* (pp. 251–255). Retrieved from www.aaai.org

van den Akker, M., Baarsma, H., Hurink, J., Modelski, M., Jan Paulus, J., Reijnen, I., ... Schreuder, J. (2008). Shunting passenger trains: getting ready for departure. *Proceedings of European Study Group Mathematics with Industry*.

Van Den Broek, R., Hoogeveen, H., Van Den Akker, M., & Huisman, B. (2021). A Local Search Algorithm for Train Unit Shunting with Service Scheduling. , 1–42.

van der Linden, J. G., Jesse Mulderij, t., Huisman, B., van den Akker, M., Han Hoogeveen, u., & Mathijs de Weerdt, u. M. (2021, 6). TORS: A Train Unit Shunting and Servicing Simulator Demonstration Track. Retrieved from https://doi.org/10.1287/trsc.1030.0076 doi: 10.5555/3463952.3464237

Van De Ven, A., Zhang, Y., Lee, W. J., Eshuis, R., & Wilbik, A. (2019). Determining capacity of shunting yards by combining graph classification with local search. *ICAART 2019 - Proceedings of the 11th International Conference on Agents and Artificial Intelligence*, *2*, 285–293. doi: 10.5220/0007398502850293

Waltz, D. (1975). Understanding Line Drawings of Scenes with Shadows. *THE PSYCHOLOGY OF COMPUTER VISION*, pages. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.2644

Whitley, D. (2001, 12). An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and Software Technology*, *43*(14), 817–831. doi: 10.1016/S0950-5849(01)00188-4

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423–1447. doi: 10.1109/5.784219

Yao, X., & Mckay, R. I. (2001). *Simulated evolution and learning: An introduction* (Vol. 15) (No. 3). doi: 10.1023/A:1011288128914

Yuen, C. C. (2004). *Selective crossover using gene dominance as an adaptive strategy for genetic programming.* London. Retrieved from https://books.google.nl/books?id=jOhKyG7WHT4C&pg=PA48&lpg=PA48&dq=Chi+Chung+Yuen,+Selective+crossover+using+gene+dominance+as+an+adaptive+strategy+for+genetic+programming.+Msc+intelligent+systems,+UK,+September+2004.&source=bl&ots=r39Z2uHsxd&sig=ACfU3U3KKgeTke5WW5DVZYrsAf-GnGcTUg&hl=nl&sa=X&ved=2ahUKEwi2h8bdiLH1AhXR-6QKHeIKDIsQ6AF6BAgCEAM#v=onepage&q=Chi%20Chung%20Yuen%2C%20Selective%20crossover%20using%20gene%20dominance%20as%20an%20adaptive%20strategy%20for%20genetic%20programming.%20Msc%20intelligent%20systems%2C%20UK%2C%20September%202004.&f=false

Yueqin, Z., Jinfeng, L., Fu, D., & Jing, R. (2007). Genetic algorithm in vehicle routing problem. *Proceedings - 3rd International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2007.*, *2*, 578–581. doi: 10.1109/IIHMSP.2007.4457776

Zhang, C., Yang, H., & Li, J. (2018, 10). Hybrid Discrete Differential Evolution Algorithm for Motor Train-Sets Scheduling. *Chinese Control Conference, CCC*, *2018-July*, 3245–3248. doi: 10.23919/CHICC.2018.8484064

Zhang, X., Ma, Z., Ding, B., Fang, W., & Qian, P. (2022). A coevolutionary algorithm based on the auxiliary population for constrained large-scale multi-objective supply chain network. *Mathematical Biosciences and Engineering*, *19*(1), 271–286. doi: 10.3934/MBE.2022014

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary computation*, *8*(2), 173–195. doi: 10.1162/106365600568202