Evolutionary algorithms for robot path planning, task allocation and collision avoidance in an automated warehouse

by

Marco Croucamp

Dissertation presented for the degree of Doctor of Philosophy
in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. Jacomine Grobler

April 2022

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

April 2022

Abstract

Research with regard to path planning, task allocation and collision avoidance is important for improving the field of warehouse automation. The dissertation addresses the topic of routing warehouse picking and binning robots. The purpose of this dissertation is to develop a single objective and multi-objective algorithm framework that can sequence products to be picked or binned, allocate the products to robots and optimise the routing through the warehouse. The sequence of the picking and binning tasks ultimately determines the total time for picking and binning all of the parts. The objectives of the algorithm framework are to minimise the total time for travelling as well as the total time idling, given the number of robots available to perform the picking and binning functions. The algorithm framework incorporates collision avoidance since the aisle width does not allow two robots to pass each other. The routing problem sets the foundation for solving the sequencing and allocation problem. The best heuristic from the routing problem is used as the strategy for routing the robots in the sequencing and allocation problem. The routing heuristics used to test the framework in this dissertation include the return heuristic, the s-shape heuristic, the midpoint heuristic and the largest gap heuristic. The metaheuristic solution strategies for single objective part sequencing and allocating problem include the covariance matrix adaptation evolution strategy (CMA-ES) algorithm, the genetic algorithm (GA), the guaranteed convergence particle swarm optimisation (GCPSO) algorithm, and the self-adaptive differential evolution algorithm with neighbourhood search (SaNSDE). The evolutionary multi-objective algorithms considered in this dissertation are the non-dominated sorting genetic algorithm III (NSGA-III), the multi-objective evolutionary algorithm based on decomposition (MOEAD), the multiple objective particle swarm optimisation (MOPSO), and the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES).

Solving the robot routing problem showed that the return routing heuristic outperformed the s-shape, largest gap and midpoint heuristics with a significant margin. The return heuristic was thus used for solving the routing of robots in the part sequencing and allocation problem.

The framework was able to create feasible real-world solutions for the part sequencing and allocation problem. The results from the single objective problem showed that the CMA-ES algorithm outperformed the other metaheuristics on the part sequencing and allocation problem. The second best performing metaheuristic was the SaNSDE.

The GA was the third best metaheuristic and the worst performing metaheuristic was the GCPSO. The multi-objective framework was able to produce feasible trade-off solutions and MOPSO was shown to be the best EMO algorithm to use for accuracy. If a large spread and number of Pareto solutions are the most important concern, MOEAD should be used.

The research contributions include the incorporation of collision avoidance in the robot routing problem when using single and multi-objective algorithms as solution strategies. This dissertation contributes to the research relating to the performance of metaheuristics and evolutionary multi-objective algorithms on routing, sequencing, and allocation problems. To the best of the author's knowledge, this dissertation is the first where these four metaheuristics and evolutionary multi-objective algorithms have been tested for solving the robot picking and binning problem, given that all collisions must be avoided. It is also the first time that this specific variation of the part sequencing and allocation problem has been solved using metaheuristics and evolutionary multi-objective algorithms, taking into account that all collisions must be avoided.

Opsomming

Navorsing in verband met roete beplanning, part allokasie en botsing vermyding is belangrik vir die bevordering van die pakhuis automatisering veld. Die verhandeling handel oor die onderwerp van parte wat gestoor en gehaal moet word en die verkillende parte moet ook gealokeer word aan 'n spesifieke robot. Die doel van hierdie verhandeling is om 'n enkele doelwit en 'n multidoelwit algoritme raamwerk te ontwikkel wat parte in 'n volgorde rangskik en ook die parte aan 'n robot alokeer. Die roete wat die robot moet volg deur die pakhuis moet ook geoptimeer word om die minste tyd te verg. Die volgorde van die parte bepaal uiteindelik die totale tyd wat dit neem vir die robot om al die parte te stoor en te gaan haal. Die doelwitte van die algoritme raamwerk is om die totale reistyd en die totale ledige tyd te minimeer, gegewe die aantal beskikbare robotte in die sisteem om die stoor en gaan haal funksies uit te voer. Die algoritme raamwerk bevat botsingsvermyding, aangesien die gangbreedte van die pakhuis nie toelaat dat twee robotte mekaar kan verbygaan nie. Die roete probleem lê die grondslag vir die oplossing van die volgorde en allokerings probleem. Die beste heuristiek vir die roete probleem word verder gebruik in die volgorde en allokerings probleem. Die verskillende roete heuristieke wat in hierdie verhandeling oorweeg was, sluit in die terugkeer heuristiek, die s-vorm heuristiek, die middelpunt heuristiek en die grootste gaping heuristiek. Die metaheuristieke vir die volgorde en allokerings probleem sluit die volgende algoritmes in: die kovariansie matriks aanpassing evolusie algoritme (CMA-ES), die genetiese algoritme (GA), die gewaarborgde konvergerende deeltjie swermoptimerings (GCPSO) algoritme, en laastens die selfaanpassende differensiële evolusie algoritme met die teenwoordigheid van buurtsoek (SaNSDE). Die evolusionêre multidoelwit algoritmes wat oorweeg was vir die volgorde en allokerings probleem sluit die volgende algoritmes in: die multidoelwit kovariansie matriks aanpassing evolusie algoritme (MO-CMA-ES), die nie-dominerende sortering genetiese algoritme III (NSGA-III), die multidoelwit evolusionêre algoritme gebaseer op ontbinding (MOEAD) en laastens die multidoelwit deeltjie swermoptimering algoritme (MOPSO).

Oplossings van die robot roete probleem het gewys dat die terugkeer heuristiek die s-vorm, grootste gaping en middelpunt heuristiek met 'n beduidende marge oortref het. Die terugkeer heuristiek is dus gebruik vir die oplossing van die roete beplanning van robotte in die volgorde en allokasie probleem.

Die raamwerk was doeltreffend en die resultate het getoon, vir die enkel doelwit probleem, dat die CMA-ES algoritme beter gevaar het as die ander metaheuristieke vir die volgorde en allokasie probleem. Die SaNSDE was die naas beste presterende metaheuristiek. Die GA was die derde beste metaheuristiek, en die metaheuristiek wat die slegste gevaar het, was die GCPSO. Vir die multidoelwit probleem het die MOPSO die beste gevaar, as akkuraatheid die belangrikste doelwit is. As 'n grootter verskeidenheid die belangrikste is, is die MOEAD meer geskik om 'n oplossing te vind.

Die navorsingsbydraes sluit in dat vermyding van botsings in ag geneem word in die robot roete probleem. Hierdie verhandeling dra by tot die navorsing oor die oplossing van roete beplanning, volgorde en allokasie probleme met metaheuristieke. Na die beste van die outeur se kennis is hierdie die eerste keer dat al vier metaheuristieke getoets was om die robot stoor-en-gaan haal probleem op te los, onder die kondisie dat alle botsings vermy moet word. Dit is ook die eerste keer dat hierdie spesifieke variant, enkel-en-multidoelwit probleem van die volgorde en allokasie van parte met behulp van metaheuristieke en multidoelwit evolusionêre algoritmes opgelos was, met die inagneming dat alle botsings vermy moet word.

Acknowledgements

The author wishes to acknowledge the following people and institutions for their contributions towards the completion of this work:

- Prof. Jacomine Grobler for always believing in me and for her excellent mentorship.
- My wife for her unconditional love and support during the completion of this dissertation.
- My family and friends for their support and encouragement during the completion of this dissertation.

Table of Contents

Contents

List of Figures

List of Figures

List of Tables

List of Tables

List of Algorithms

List of Algorithms

List of Acronyms

AGV	Automated guided vehicle
AGVS	Automated guided vehicle system
APF	Artificial potential function
CMA-ES	Covariance matrix adaptation evolution strategy
DE	Differential evolution
DH	Dedicated heuristic
EMO	Evolutionary multi-objective optimisation
ES	Evolution strategies
GA	Genetic algorithm
GCPSO	Guaranteed convergence particle swarm optimisation
GMM	Gaussian mixture model
HGA	Hybrid genetic algorithm
HV	Hypervolume
JIT	Just-in-time
JPS	Jump point search
MO-CMA-ES	Multi-objective covariance matrix adaptation evolution strategy
MOEAD	Multi-objective evolutionary algorithm based on decomposition
MOO	Multi-objective optimisation
MOPSO	Multi-objective particle swarm optimisation
NPS	Number of Pareto solutions
NSDE	Differential evolution with neighbourhood search
NSGA-III	Non-dominated sorting genetic algorithm
PAES	Pareto archive evolution strategy
PSO	Particle swarm optimisation
RRT	Rapidly exploring random tree
SaDE	Self-adaptive differential evolution
SaNSDE	Self-adaptive differential evolution algorithm with neighbourhood search
SM	Spacing metric
SPLOM	Scatter plot matrices
SSD	Solution space diversity
STAMC	Simultaneous task allocation and motion coordination
STSP	Steiner travelling salesman problem
TSP	Travelling salesman problem

List of Reserved Symbols

$\mathbf{a} \triangleq$ set of parts allocated to a robot.

$a \triangleq$ number of aisles.

$\mathbf{b} \triangleq$ set of parts to be binned per robot.

$B^{(g)} \triangleq$ represents an orthogonal $n \times n$ matrix.

$C(\pi) \triangleq$ the value or cost of the permutation.

$c_i'(t) \triangleq$ i^{th} offspring at time t.

$c_{ij}(t) \triangleq$ denotes the j^{th} dimensions of the i^{th} offspring solution.

$c_{1/2} \triangleq$ the cognitive and social acceleration constants respectively.

$C_t \triangleq$ set of items contained in a list t.

$\boldsymbol{D}^{(t)} \triangleq$ a diagonal $n_x \times n_x$ matrix obtained from the eigen decomposition.

$D_{k,l} \triangleq$ distance between a location k and l.

$f_{\delta l} \triangleq$ improvement in fitness value.

$\boldsymbol{F_l} \triangleq$ last front.

$\boldsymbol{f}(\boldsymbol{x}) \triangleq$ denotes the vector of the objective function to be minimised.

$\gamma \triangleq$ a linear or exponentially decreasing value.

$g \triangleq$ the generation number.

$g \triangleq$ the generation number.

$I \triangleq$ number of parts in the warehouse.

$I \triangleq$ the number of parts in the warehouse.

$K \triangleq$ the number of robots in the system.

$\lambda \triangleq$ population size.

$m \triangleq$ number of picking lists.

$m_{i,k} \triangleq$ ending time at point j for robot k.

$n \triangleq$ number of items (and locations).

$n_{i,k} \triangleq$ starting time of new task for part i from the entrance gate.

$N_i(0.5; 0.3) \triangleq$ a Gaussian random number with mean 0.5 and standard deviation 0.3.

$NM(0, \boldsymbol{M}) \triangleq$ denotes the multivariate normal distribution.

$ns \triangleq$ is the population size.

$ns_{1/2} \triangleq$ the number of offspring generated.

$n_x \triangleq$ the number of dimensions.

$\mathbf{p} \triangleq$ set of parts to be picked per robot.

$p \triangleq$ process time for picking or binning incurred for all parts.

$p_c \triangleq$ the crossover probability.

$\boldsymbol{PF} \triangleq$ Pareto optimal set.

$p_T \triangleq$ probability of reproduction.

$\boldsymbol{P_t} \triangleq$ parent population.

$\boldsymbol{Q_t} \triangleq$ the offspring of $\boldsymbol{P_t}$.

$r_{1j}(t) \triangleq$ random sample from uniform random distribution $U(0, 1)$.

$r_{2j}(t) \triangleq$ random sample from uniform random distribution $U(0, 1)$.

$\boldsymbol{\rho} \triangleq$ niche count.

$s \triangleq$ current state.

$s^* \triangleq$ neighbouring state.

$t_i \triangleq$ the transport time without delays of part i from entrance to exit of warehouse.

$\boldsymbol{T}_{ij}(t) \triangleq$ target vector.

$U \triangleq$ the number of best individuals that will be recombined.

$u_{i,t} \triangleq$ picking order of item i in list t.

$\boldsymbol{\varrho_q} \triangleq$ are the equality constraints.

$v_{ij}(t) \triangleq$ the velocity of particle i in dimension j at time t.

$\varsigma_{ij}(t) \triangleq$ is referred to as the standard deviation of the Gaussian mutations.

$w \triangleq$ inertia weight.

$w_i \triangleq$ recombination of weights.

$w_{i,j,k} \triangleq$ time added to avoid collision if part i is picked or binned.

$w_l \triangleq$ weight contribution to l.

$\mathbf{x} \triangleq$ set of continuous variables given by the metaheuristic where $x = 3 \times I$.

$\widehat{x}_{ij}/x_{ij}(t) \triangleq$ the *pbest* and position of a particle i in dimension j at time t respectively.

$x_{\tau j}(t) \triangleq$ the j^{th} component of the best individual in the population at time t.

$x_j^*(t) \triangleq$ global best position (*gbest*) in dimension j.

$x_k^{t+1} \triangleq$ represents the offspring created.

$\boldsymbol{x}_{p1}(t) \triangleq$ parent individual.

$\boldsymbol{x}_{p2}(t) \triangleq$ parent individual.

$x_{pkj}(t) \triangleq$ denotes the j^{th} dimension of the k^{th} vector of the parent individual i of generation t.

$x_w^{(g)} \triangleq$ the generation number.

$\boldsymbol{Z}^r \triangleq$ supplied reference points.

$\boldsymbol{\zeta_q} \triangleq$ are the inequality constraints.

Chapter 1

Introduction and background

This chapter introduces the real-world warehouse considered in this dissertation and describes its background. Section 1.1 discusses the background of the warehouse routing problem and the importance of warehousing activities. Section 1.2 discusses the contributions that this dissertation is aiming to contribute. The research objectives are discussed in Section 1.3, followed by the dissertation structure in Section 1.4.

1.1 Background

Warehouse efficiency has a significant impact on a supply chain's efficiency. The efficiency of a supply chain is measured by the time and costs of warehouse activities, amongst other factors. The benefits of optimising warehouse activities include a decrease in order picking and binning time, cost savings on correct part picking, a decrease in incorrect shipments and an overall increase in customer satisfaction. The presence of an optimised warehouse system has a directly proportional positive impact on the competitiveness of a business's supply chain. Specifically efficient and effective warehousing in the supply chain can lead to greater competitiveness. The competitive advantages are customer service, decreased lead times, and decreased logistical costs.

The goal to achieve a lean and efficient warehouse has led to an increasing interest in warehouse functions. This field has been studied in both literature and industrial practice. Warehouse functions have become a critical part of business supply chains. Warehouse functions and operations include, but are not limited to, the receiving, storing and dispatching of products. Part allocation and batching are important planning and operational factors when considering optimising the warehouse routing strategy (De Koster et al., 2007a).

The costs of warehouse activities also need to be considered. Costs related to order picking can be as high as 45 to 55% of all operational costs (Tompkins et al., 1996). Although this is not a new concept, businesses struggle with the implementation of automatic warehouse systems due to inexperience, capital expenses and technological gaps. Automated warehousing solutions have been conceptualised since 2009 (Banker, 2009). In 2009 Kiva robots were introduced by Amazon, but nobody knew that it would eventually change the face of automated warehousing. Amazon, for example, is using over 30 000 of the Kiva robots in their facilities to perform picking and binning tasks (Banker, 2009).

The purpose of this dissertation is to develop an algorithm framework that is able to use metaheuristics and evolutionary multi-objective algorithms to solve the part sequencing and allocation problem, and optimise the routing strategy through the warehouse. The sequence of the picking and binning tasks ultimately determines the total time required for the picking and binning. The main objectives of the framework includes minimising the total time needed to pick and bin parts and minimise the idle time for all robots. The algorithm framework contributes significantly to literature because it incorporates collision avoidance.

The performance of the metaheuristics and other algorithms are tested in the part sequencing and allocation framework. This dissertation focuses on the performance of the metaheuristics and evolutionary multi-objective algorithms on the part sequencing and allocation problem. The metaheuristics considered were the covariance matrix adaptation evolution strategy, genetic algorithm, particle swarm optimisation, and differential evolution strategies. The evolutionary based multi-objective algorithms include the non-dominated sorting genetic algorithm III, the multi-objective evolutionary algorithm based on decomposition, the multiple objective particle swarm optimisation, and the multi-objective covariance matrix adaptation evolution strategy. In addition to the metaheuristics and the evolutionary multi-objective algorithms listed, different routing heuristics are tested to solve the robot routing problem. These heuristics include the midpoint heuristic, the s-shape heuristic, the return heuristic, and the largest gap heuristic.

The robot routing problem is solved before the sequencing and allocation problem because the best performing routing heuristic is used in the part sequencing and allocation algorithm framework. For solving the single objective or multi-objective part sequencing and allocation problem the framework is set up over five data sets with 30 simulation runs each. The five data sets included the following problem sizes: 8 parts, 16 parts, 24 parts, 32 parts, and 40 parts.

The warehouse considered in this dissertation is used as a buffer storage area to keep stock for a just-in-time (JIT) supply chain. The warehouse currently stocks 40 parts. The binning and picking of products in the warehouse happens daily. Currently, the warehouse tasks are done manually by personnel using a forklift or by hand. The personnel and forklift driver are responsible for collecting and replenishing all the products required and the dispatching of products from the warehouse to the assembly floor is a continuous process. The simplified floor plan of the warehouse in the assembly plant can be seen in Figure 1.

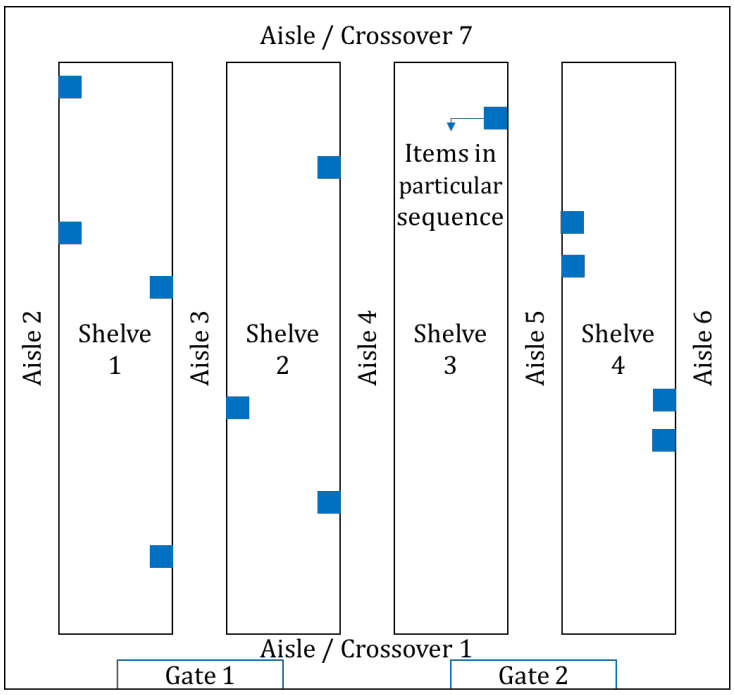

Figure 1: Warehouse floor plan

The warehouse consists of pallet racks with parallel aisles between the racks. There are two cross aisles at the front and back of the warehouse, respectively, and there are two gates allowing access into and out of the warehouse. One gate is used as an entrance and the other as an exit. All products have a designated storage space allocated to them and the space allocation does not change frequently.

The operation of order picking and binning are considered to be similar in this dissertation. The robot has to move from the entrance gate to the part's assigned destination to either pick or bin a part. The picking or binning operations refer to the collection or storage of parts, respectively. The picking and binning quantities are based on the customer's demands and the supplier's delivery schedule (Roodbergen and De Koster, 2001).

Total picking time refers to the actual time it takes to complete all the activities required for picking an order. These activities include: travel time from the current position of the robot to the offloading or loading position, searching for the part to be picked, and the actual picking time (Dekker et al., 2004). De Koster et al. (2007b) identified four methods to possibly reduce travelling distances, namely:

1. Storage location assignment. Storage location refers to the fixed position where a part is stored. This position does not change frequently.

2. Warehouse zoning. Warehouse zoning has been used in a larger warehouse where the warehouse is divided into zones, since the zones help with finding locations more easily.

3. Order batching. Batching includes the collection of multiple items in one trip.

4. Vehicle routing methods. Vehicle routing methods are used to guide the vehicle from start to finish, and includes the path that the vehicle will follow to successfully complete the picking of orders and reaching the end destination.

For this dissertation, the framework sequences all of the products to be picked or binned and allocates each part to a respective robot. A part can only be allocated to one robot for picking or binning. The robot uses the sequence of parts given by the algorithm framework to complete the picking and binning. Furthermore, the robots use optimised routes that are collision free. The next section includes the detailed problem statement.

The aim can be formulated as follows:

Developing an algorithm framework that can efficiently identify parts to be picked or binned, assign them to a robot, and create the best found picking and binning sequence. The objectives are to minimise the total time required to pick and bin all of the parts and minimise the total idle time for all robots. The robot routing algorithm needs to incorporate collision avoidance since the aisle width does not allow two robots to pass each other in the same aisle.

1.2 Contributions

The research contributions of this dissertation include the incorporation of collision avoidance in the robot routing and path planning problem while using metaheuristics and evolutionary multi-objective algorithms as solution strategies. This dissertation contributes to the research into the performance of metaheuristics routing, multi-objective algorithms, collision avoidance, and sequencing and allocation problems. To the best of the author's knowledge, this dissertation is the first where these four metaheuristics and evolutionary multi-objective algorithms have been tested for solving the robot picking and binning problem, given that all collisions must be avoided. Finally, a framework is developed for solving the part sequencing and allocation problem. The framework addresses multiple robots, path planning, task allocation and collision avoidance. To the best of the author's knowledge this specific problem scenario cannot be found in literature.

1.3 Research objectives

The main objective of this dissertation is to develop a framework that is able to find feasible solutions for the single and multi-objective part sequencing and allocation problem. This objective is divided into sub-objectives which will lead up to the completion of the main objective. The sub-objectives are listed, along with the chapter that addresses the sub-objective.

1. Provide feasible robot routing solutions for the robot routing problem - Chapter 2
2. Identify and discuss evolutionary single objective and multi-objective algorithms that are used to solve the single and multi-objective part sequencing and allocation problem - Chapter 2
3. Investigate and determine which metrics to consider for analysing the algorithm results - Chapter 2
4. Motivate the necessity for efficient and automated warehousing solutions - Chapter 3
5. Provide a detailed review of routing and path planning strategies for both single and multi-objective path planning and task allocation problems - Chapter 3
6. Develop a mathematical model and algorithm framework for the single objective part sequencing and allocation problem - Chapter 4
7. Develop an experimental evaluation to determine the best routing strategy and discuss the results - Chapter 5

8. Develop a single objective algorithm framework to solve the single objective part sequencing and allocation problem - Chapter 6
9. Evaluate the performance of the single objective algorithm framework - Chapter 6
10. Identify objective functions that can be used in the multi-objective algorithm framework that is not correlated - Chapter 7
11. Develop a multi-objective algorithm framework to solve the part sequencing and allocation problem - Chapter 8
12. Setup and evaluate the performance of the multi-objective algorithm framework on the multi-objective part sequencing and allocation problem - Chapter 9
13. Conclude the results obtained from the algorithm framework and provide suggestions for future research - Chapter 10

1.4 Dissertation structure

Chapter 2 provides feasible solution strategies for solving the robot routing problem. Also discussed in chapter 2 are the evolutionary single objective and multi-objective algorithms that are used for solving the part sequencing and allocation problem. The metrics used for analysing the framework results are also discussed in chapter 2. Chapter 3 explores literature available and discusses automated warehouse systems, warehouse layouts, warehousing activities, automated warehouse routing, optimisation algorithms, the travelling salesman problem, routing an order picking or binning robot, and the sequencing of parts. Chapter 3 also gives a detailed review of routing strategies for the single and multi-objective path planning and task allocation problem. Chapter 4 develops and describe the mathematical model and the part sequencing and allocation algorithm framework for the single objective part sequencing and allocation problem. Chapter 5 presents the data analysis and empirical evaluation results for the robot routing problem. Chapter 6 discusses the single objective part sequencing and allocation problem results. Chapter 7 identifies the objectives that are used for the multi-objective algorithm framework. Chapter 8 develops the multi-objective algorithm framework, using the identified objective functions from chapter 7. Chapter 9 develops and evaluates the performance of the multi-objective algorithm framework on the part sequencing and allocation problem using the metrics investigated in chapter 2. Chapter 10 concludes the dissertation with a summing up and topics for future work.

Chapter 2

Metaheuristic solution strategies for robot picking

Heuristics are often defined for a given problem. These heuristics can be problem-specific and do not guarantee an optimal solution. Heuristics are normally strategies derived from previous experiences with similar problems and usually need a well-defined problem with a given set of rules. Metaheuristics tend to be problem-independent techniques. Different metaheuristics can be applied to multiple problems. A heuristic can be as simple as choosing a random solution from a list whereas a metaheuristic does not have to have any information about the problem; it can treat fitness functions as black boxes. This chapter is structured as follows: Section 2.1 discusses various single objective optimisation algorithms including the genetic algorithm, the differential evolution algorithm, the covariance matrix adaptation evolution strategy (CMA-ES) and the particle swarm optimisation (PSO) algorithm. Section 2.2 discusses a number of multi-objective optimisation algorithms including the non-dominated sorting genetic algorithm III (NSGA-III), the multi-objective evolutionary algorithm based on decomposition (MOEAD), the multiple objective particle swarm optimisation (MOPSO), and the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES).

2.1 Single objective optimisation algorithms

Optimisation algorithms solve problems by testing different combinations of input variables in an attempt to find an acceptable solution. These algorithms use different strategies to find an acceptable solution, such as crossover mechanisms, mutation processes and other forms of generating the next set of candidate solutions. These strategies are used in an iterative manner, where in each iteration the candidate solutions created are tested and evaluated based on a fitness function. The optimisation algorithm attempts to improve the best fitness values until no better solution can be found or when the terminating condition is met.

Figure 2 (Grobler et al., 2008) shows the interrelationship between the commonly used optimisation algorithms. The first distinction is between continuous and combinatorial problems. Exact methods can take a significant amount of time to solve. Therefore, to find a near-optimal solution in a smaller time frame an approximate method would be more useful. Approximate methods for a combinatorial problem include metaheuristics and heuristics.

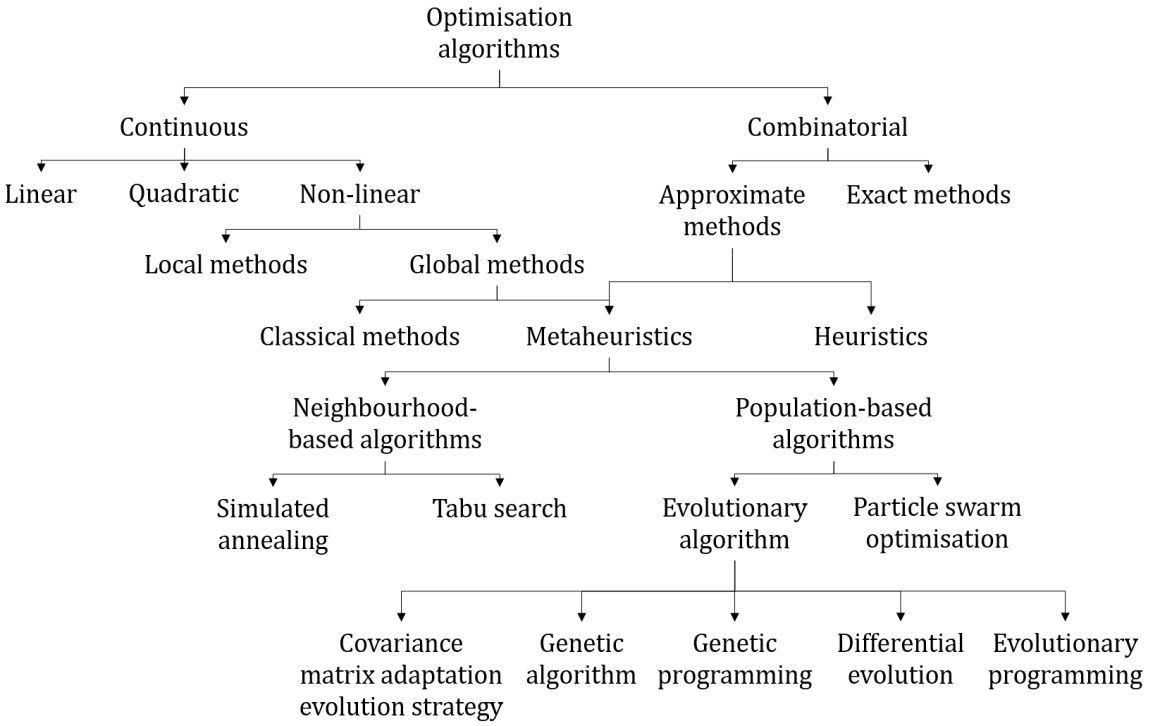

Figure 2: Optimisation algorithms (Grobler et al., 2008)

In the case where the problem is continuous in nature, linear, quadratic, or non-linear methods can be used to find a candidate or near-optimal solution. Non-linear problems can be solved using local methods, searching for local optimums, or global methods. Global methods searches a very large solution space to find the best possible solution given the termination criteria. Global methods include classical methods and metaheuristics.

Metaheuristics can be subdivided into two main groups, neighbourhood-based algorithms and population-based algorithms. Neighbourhood-based algorithms include simulated annealing and tabu search. For simulated annealing, consider a neighbouring state $\boldsymbol{s}^*$ of the current state $\boldsymbol{s}$. A probabilistic process decides to move the system to a state $\boldsymbol{s}^*$ or remain in state $\boldsymbol{s}$. These probabilities lead the system to move to a state that is of sufficient quality for the application or until a termination criterion is met. The tabu search's strategy is to move iteratively from one potential solution x to a better solution $\boldsymbol{x}^*$ in the neighbourhood of $\boldsymbol{x}$, until a stopping criterion has been satisfied. The tabu search adopts the better solution and stores past solutions in a tabu list. It is possible for the tabu search to adopt a poor solution to escape out of a local optimum. The tabu search thus has the ability to eliminate the risk of following the same route by verifying that the better solution found is a new solution and is not part of the current tabu list.

A typical population-based algorithm is the particle swarm optimisation (PSO) algorithm. The PSO algorithm assigns a velocity to each individual or candidate solution. The individual then moves through hyperspace searching for better solutions at the assigned velocity.

The evolutionary algorithms include the covariance matrix adaptation evolution strategy (CMA-ES), the genetic algorithm (GA), genetic programming, differential evolution (DE) and evolutionary programming. The CMA-ES learns the relationships and dependencies between a given set of decision variables by adapting a covariance matrix. The genetic algorithm relies on biological operations such as mutation, crossover and selection to find the best possible solution in the solution space. The differential evolution aims to improve the population by means of recombination, evaluation and selection during each iteration.

The following sections introduce the metaheuristics that were used to solve the single objective part sequencing and allocation problem. Section 2.1.2 discusses the genetic algorithm used in this dissertation. Section 2.1.3 describes the basic differential evolution (DE) algorithm as well as the improved self-adaptive differential evolution algorithm (SaNSDE). Section 2.1.4 explains the covariance matrix adaptation evolution strategy (CMA-ES) algorithm in detail. Section 2.1.5 presents the basic particle swarm optimisation (PSO) algorithm and also the improved guaranteed convergence particle swarm optimisation (GCPSO).

2.1.1 Single objective robot routing literature

Dekker et al. (2004) modified existing solution techniques in order to solve the order picker routing problem. The existing solution strategies considered by Dekker et al. (2004) include: the midpoint heuristic, the largest gap heuristic, s-shape heuristic and combined heuristic. The four methods listed above, including the return heuristic, are discussed in more depth in the rest of this section. See Figure 3 for examples of these routing heuristics.

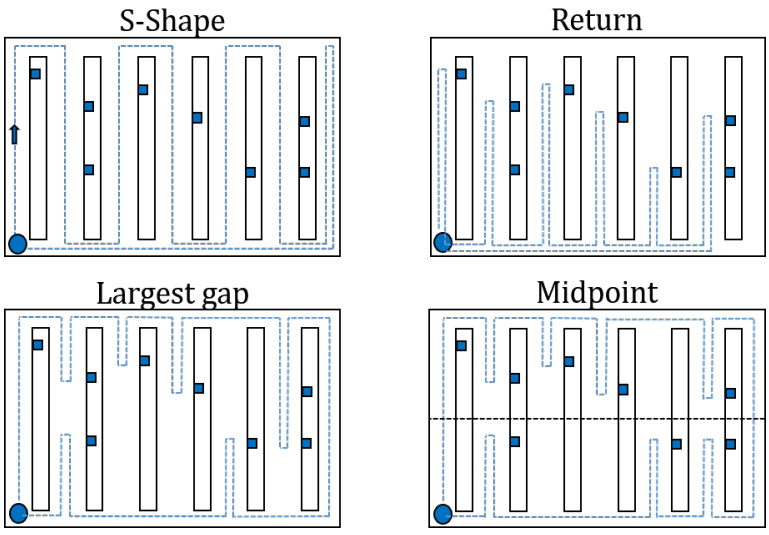

Figure 3: Routing heuristics examples

Midpoint heuristic

The midpoint heuristic commences by separating the warehouse into two horizontal sections, creating a hypothetical midline between them (Theys et al., 2007). The midpoint heuristic then allows for all the products in section one to be picked. After all the products in section one have been picked the robot can move into the second section. The pseudocode for the midpoint heuristic is shown in Algorithm 1:

Algorithm 1: Midpoint heuristic

```
Initialise a warehouse with I number of storage spaces per aisle and a number of aisles
Midpoint = I/2
for aisle_number ≤ a do
    if there is a part to be picked in an allocation coordinate ≤ Midpoint then
        Pick all parts in the aisle ≤ Midpoint
    else
        Skip to next aisle (a = a + 1)
end
Move to the opposite side of the warehouse
a = 0
for aisle_number ≤ a do
    if there is a part to be picked in an allocation coordinate ≥ Midpoint then
        Pick all parts in the aisle ≥ Midpoint
    else
        Skip to next aisle(a = a + 1)
end
```

Largest gap

The largest gap heuristic follows the same methodology as the midpoint heuristic. The only difference is that the robot follows an aisle until it reaches the largest gap in the aisle. The robot then returns to the beginning of the aisle and moves on to the next aisle. The largest gap is the largest part of an aisle where no parts need to be picked. The pseudocode for the largest gap heuristic can be seen in Algorithm 2.

Algorithm 2: Largest gap heuristic

```
Initialise a warehouse with I number of storage spaces per aisle and a number of aisles
for aisle_number ≤ a do
    define Largest_gap for aisle_number ≤ a
    if there is a part to be picked in an allocation coordinate ≤ Largest_gap then
        Pick all parts in the aisle ≤ Largest_gap
    else
        Skip to next aisle (a = a + 1)
end
Move to the opposite side of the warehouse
a = 0
for aisle_number ≤ a do
    if there is a part to be picked in an allocation coordinate ≥ Largest_gap then
        Pick all parts in the aisle ≥ Largest_gap
    else
        Skip to next aisle (a = a + 1)
end
```

S-shape

Hall (1993) commented that the s-shape heuristic is significantly different from the midpoint and largest gap heuristics. In the s-shape heuristic, the robot only enters aisles which have a part to be picked. For each aisle the robot enters, it travels through the whole aisle; in other words the robot enters the aisle at one point and exits at the opposite point of the aisle.

Aisles that have no order to be picked are skipped (Dukic and Oluic, 2007). After picking the part in a specific aisle the robot drives through the whole aisle and returns to the main aisle in the form of an 's', until all the products have been picked. For clarification, the pseudocode is shown in Algorithm 3.

Algorithm 3: S-shape heuristic

```
Initialise a warehouse with I number of storage spaces per aisle and a number of aisles
for aisle_number ≤ a do
    if there is a part to be picked in aisle a then
        Turn into aisle a and move through the whole aisle
    else
        Skip to next aisle(a = a + 1)
end
```

Combined

The combined heuristic follows the same methodology as the s-shape heuristic. The combined heuristic uses both the s-shape heuristic for main route planning and when inside an aisle, the combined heuristic uses the largest gap methodology to look ahead. Roodbergen and Koster (2001) described this algorithm in depth. The pseudocode for the combined heuristic can be seen in Algorithm 4.

Algorithm 4: Combined heuristic

```
Initialise a warehouse with I number of storage spaces per aisle and i number of aisles
Define Largest_gap for aisle_number ≤ a
for aisle_number ≤ a do
    if there is a part to be picked in aisle a then
        if there is a part to be picked in an allocation coordinates ≤ Largest_gap then
            Pick all parts in the aisle which are ≤ Largest_gap
        else
            Skip to next aisle (a = a + 1)
    else
        there is no part to be picked in aisle a
end
Move to the opposite side of the warehouse
for aisle_number ≤ a do
    if there is a part to be picked in aisle a then
        if there is a part to be picked in allocation coordinates ≥ Largest_gap then
            Pick all parts in the aisle which are leq Largest_gap
        else
            Skip to next aisle (a = a + 1)
    else
        There is no part to be picked in aisle a
end
```

Return

The last order picking heuristic is called the return heuristic. The return heuristic does exactly what the name describes. The robot will enter every aisle, collect all the items in the aisle and return to the beginning of that aisle. If there are no products to be picked in an aisle the robot will skip that aisle.

In this heuristic, a robot will never transverse any aisle completely (Dukic and Oluic, 2007). The pseudocode for the return heuristic is shown in Algorithm 5.

Algorithm 5: Return heuristic

```
Initialise a warehouse with I number of storage spaces per aisle and a number of aisles
for aisle_number ≤ a do
    if there is a part to be picked in aisle a then
        Turn into aisle a and pick all the parts in this aisle
        return to point of entry for aisle a
    else
        Skip to next aisle (a = a + 1)
end
```

All the routing heuristics described in the paragraphs above are modifications or combinations of existing heuristics, but none of them optimises the sequencing of the order picking. The sequencing of the order picking products must be done by more intelligent metaheuristics to obtain higher quality solutions.

2.1.2 Genetic algorithm

The GA is a population-based optimisation algorithm. A population can be defined as a group of potential individual solutions. These populations are created and reproduced by previous populations by using crossover and mutation of individuals.

The GA is inspired by the process of natural selection using biological operations such as mutation, crossover and selection (Mitchell, 1996), (Sadeghi et al., 2014). GAs are commonly used to find high quality solutions for optimisation and search related problems. John Holland introduced the GA in the 1960s and after its invention the GA was popularised by David Goldberg (Goldberg and Holland, 1988).

In this dissertation the GA with floating point representation, tournament selection, blend crossover (Eshelman and Schaffer, 1993) and Gaussian mutation was used (Olorunda and Engelbrecht, 2009). For each individual i, the two parent vectors are selected by means of tournament selection. The two parents $\boldsymbol{x}_{p1}(t)$ and $\boldsymbol{x}_{p2}(t)$, where $\boldsymbol{x}_{pkj}(t)$ denotes the j^{th} dimension of the k^{th} vector of the parent individual i of generation t where $i \neq p_1 \neq p_2$. For all dimensions, j, if $r \sim U(0;1) \leq p_c$, where p_c denotes the crossover probability, $c_{ij}(t)$ denotes the j^{th} dimensions of the i^{th} offspring solution, where $x_{p1j}(t) \leq x_{p2j}(t)$. The uniform and Gaussian distributions between 0 and 1 are denoted as $U(0;1)$ and $N(0;1)$ respectively:

$$\delta_j = 2U(0,1) - 0.5 \tag{1}$$

and

$$c_{ij}(t) = (1 - \delta_j)x_{p1j}(t) + \delta_j x_{p2j}(t) \tag{2}$$

The i^{th} offspring at time t, $c'_{ij}(t)$ can be calculated using Equation (3). ς is referred to as the standard deviation of the Gaussian mutations,

$$c'_{ij}(t) = c_{ij}(t) + \varsigma_{ij}(t)N_{ij}(0,1), \tag{3}$$

with

$$\varsigma_{ij}(t+1) = \varsigma_{ij}(t)e^{\tau_1 N(0,1) + \tau_2 N(0,1)} \tag{4}$$

$$\tau_1 = \frac{1}{\sqrt{\sqrt[2]{n_x}}} \tag{5}$$

and

$$\tau_2 = \frac{1}{\sqrt{2n_x}} \tag{6}$$

where n_x represents the number of dimensions. If the fitness of $\boldsymbol{c}'_i(t)$ is better than the original $\boldsymbol{x}_i(t)$, then the individual gets replaced by $\boldsymbol{c}'_i(t)$ (Engelbrecht, 2006).

2.1.3 Differential evolution

The DE algorithm was originally developed from work done by Storn (1996) and Storn and Price (1997). The strategy and procedure of a DE is described below as presented in Brownlee (2011). DE is a stochastic direct search and a global optimisation algorithm. The DE is an evolutionary algorithm from the field of evolutionary computation. It shares similarities with the GA, evolutionary programming, evolution strategies and there are even similarities to the PSO.

The strategy of the DE involves maintaining a population of possible solutions. These populations are evolved by means of recombination, evaluation and selection at each iteration. The recombination process creates new candidate solution components based on the weighted difference of two random population candidates added together, forming the third population candidate. This process perturbs population members. In parallel, the perturbation effect organises the sampling of the solution space, bounding it to known areas and areas that are of interest with regard to the possible optimal solution.

Grobler et al. (2008) investigated the basic DE algorithm and variations of it. The alternative DE algorithms include changes to the selection mechanisms, selection strategies and crossover mechanisms. Since the publication of the DE algorithm, researchers have studied the practical aspect of the DE in multiple optimisation problems.

The basic DE algorithm as described in the following paragraphs is discussed in greater depth in Storn (1996) and Storn and Price (1997). For each individual candidate i in the solution population, a vector $\boldsymbol{x}_{i_1}(t)$ exists. There are two other vectors $\boldsymbol{x}_{i_2}(t)$ and $\boldsymbol{x}_{i_3}(t)$ which are randomly selected from the current population where $\boldsymbol{x}_{ij}(t)$ denotes the j^{th} dimension of the candidate i of generation t and $i \neq i_1 \neq i_2 \neq i_3$. The target vector, $\boldsymbol{T}_{ij}(t)$, can be calculated using the differential mutation operator seen in Equation (7), where F is the scaling factor and $\boldsymbol{c}_i$ is known as the trial vector.

$$T_{ij}(t) = x_{i_1 j}(t) + F(x_{i_2 j}(t) - x_{i_3 j}(t)) \tag{7}$$

Then for all dimensions j, if $r \sim U(0,1) \leq p_r$ or $j = v \sim U(1, ..., n_x)$ then:

$$\boldsymbol{C}_{ij}(t) = T_{ij}(t). \tag{8}$$

Otherwise $\boldsymbol{c}_{ij}(t) = \boldsymbol{x}_{ij}(t)$, where p_r represents the probability of reproduction.

The rule of replacement states that one candidate may only replace another candidate if the fitness value of the i^{th} candidate is better than the candidate from the original population. Grobler et al. (2008) provided the pseudocode for the basic DE algorithm, reproduced in Algorithm 6.

Algorithm 6: Basic DE algorithm

Initialise an n_x-dimensional population of n_s candidates
$t = 1, i_1 = 0, i_2 = 0, i_3 = 0,$
while $t < I_{max}$ **do**
  for *each candidate* **do**
    Randomly select a candidate i_1 from the population
    Randomly select a candidate i_2 from the population
    while $i_1 = i_2$ **do**
      Randomly select a candidate i_2 from the population
    end
    Randomly select a candidate i_3 from the population
    while $i_2 = i_3$ *or* $i_1 = i_3$ **do**
      Randomly select a candidate i_3 from the population
    end
    Randomly select a dimension, v
    for *All dimensions* j **do**
      if $r \sim (0, 1) \leq p_r$ *or* $j = v$ **then**
        calculate $c_{ij}(t)$ using $F_i(t) = F_{i_4}(t) + N(0, 0.5)(F_{i_5}(t) - F_{i_6}(t))$
      else if then
        $c_{ij}(t) = x_{ij}(t)$
    end
  end
  for *each candidate* **do**
    if $f(\boldsymbol{c}_i(t)) \leq f(\boldsymbol{x}_i(t))$ **then**
      $\boldsymbol{x}_i(t + 1) = \boldsymbol{c}_i(t)$
  end
  $t = t + 1$
end

A number of self-adaptive differential evolution (SaDE) algorithms have been developed. The algorithms developed include the differential evolution algorithm with neighbourhood search (NSDE) as discussed in Yang et al. (2007), the self-adaptive differential evolution algorithm as discussed in Qin and Suganthan (2005), and the self-adaptive differential evolution algorithm with neighbourhood search (SaNSDE) (Yang et al., 2008), which combines the best features of the NSDE and the SaDE.

The SaNSDE has been shown to outperform both the SaDE and the NSDE (Yang et al., 2008). The SaNSDE is considered to be a highly successful DE algorithm and thus the SaNSDE is used in this dissertation. The pseudocode for the SaNSDE is provided in Algorithm 7.

$N_i(0.5; 0.3)$ is a Gaussian random number with mean 0.5 and standard deviation 0.3. $x_{\tau j}(t)$, denotes the j^{th} component of the best individual in the population at time t and γ denotes a linear or exponentially decreasing value, ($\gamma \in (0, 1)$). Equation (9) is used to calculate the probability of reproduction for each individual i. All of the p_{ri} values that were successful from offspring $\boldsymbol{q}_l$ are stored in a set $\boldsymbol{p}_{r_{succ}}$; and Equation (10) calculates the sum total of all the successful values.

The improvement in fitness value, $f_{\delta l}$, is calculated using Equation (14). The l^{th} weight is calculated using Equation (13). The probability p_T is calculated using Equation (12) and ns_1 denotes the number of offspring generated by Equation (7). Similarly, ns_2 denotes the number of successful offspring generated by Equation (11). nf_1 and nf_2 denote the number of offspring generated by Equations (7) and (11) (Grobler, 2015).

$$p_{ri} = Ni(p_{r_\mu}; 0:1) \tag{9}$$

$$p_{r_\mu} = \sum_{l=1}^{|p_{r_{succ}}|} w_l q_l \tag{10}$$

$$T_{ij}(t) = \gamma x_{\tau j}(t) + (1-\gamma) x_{i_1 j}(t) + F(x_{i_2 j}(t) - x_{i_3 j}(t)) \tag{11}$$

$$p_T = \frac{ns_1(ns_2 + nf_2)}{ns_2(ns_1 + nf_1) + ns_1(ns_2 + nf_2)} \tag{12}$$

with

$$w_l = \frac{f_{\delta l}}{\sum_{l=1}^{|f_\delta|} f_{\delta l}} \tag{13}$$

given

$$f_{\delta l} = f(x_l)(t) - f(c_l)(t+1) \tag{14}$$

Algorithm 7: SaNSDE algorithm

Initialise an n_x-dimensional population of n_s candidates
$t = 1$
while *no stopping condition is satisfied* **do**
  for *each individual* i **do**
    Generate a scale factor $F_i = N_i(0:5; 0:3)$
    Generate a probability factor $p_{ri} = Ni(p_{r_\mu}; 0:1)$
    Randomly select a candidate i_1 from the population
    Randomly select a candidate i_2 from the population
    while $i_1 = i_2$ **do**
      Randomly select a candidate i_2 from the population
    end
    Randomly select a candidate i_3 from the population
    while $i_2 = i_3$ *or* $i_1 = i_3$ **do**
      Randomly select a candidate i_3 from the population
    end
    $$T_{ij}(t) = \begin{cases} x_{i_1 j}(t) + F(x_{i_2 j}(t) - x_{i_3 j}(t)), & \text{if } U_i(0,1) < p_T \\ \gamma x_{\tau j}(t) + (1-\gamma)x_{i_1 j}(t) + F(x_{i_2 j}(t) - x_{i_3 j}(t)), & \text{otherwise} \end{cases}$$
    $$c_{ij}(t) = \begin{cases} T_{ij}(t), & \text{if } r \sim U(0,1) \leq p_r \text{or} j = v \sim U(1,...,n_x) \\ x_{ij}(t), & \text{otherwise} \end{cases}$$
  end
  for *each candidate* **do**
    if $f(\boldsymbol{c}_i(t)) \leq f(\boldsymbol{x}_i(t))$ **then**
      $\boldsymbol{x}_i(t+1) = \boldsymbol{c}_i(t)$
  end
  $t = t + 1$
  Update the probabilities p_T, p_f, and p_{r_μ}
end

2.1.4 CMA-ES

The covariance matrix adaptation evolution strategy (CMA-ES) algorithm is an evolutionary strategy and is designed to learn the relationships and dependencies between a given set of decision variables by adapting a covariance matrix (Hansen and Ostermeier, 1996). The matrix defines the sampling distribution of each candidate solution (Hansen et al., 2003). The capability of learning the dependencies between a number of decision variables forms the CMA-ES's practical limitations: memory storage and computational complexity per function evaluation (Ros and Hansen, 2008).

The CMA-ES is described in depth in Hansen and Kern (2004). They tested the performance of the CMA-ES on various techniques and multimodal test functions. In particular, the effect of population size was investigated. The results concluded that an increase in population size improved the performance of the CMA-ES model on six of the eight test functions. In comparison with state-of-the-art search strategies, the CMA-ES achieved superior performance on multimodal test functions. The CMA-ES benefits from larger population sizes when using ranking and weighted recombination of offspring (Ros and Hansen, 2008).

The offspring generating the next generation $(t+1)$ are sampled using:

$$\boldsymbol{x}_k^{t+1} \sim (x)_w^{(t)} + \sigma^{(t)} \boldsymbol{B}^{(t)} \boldsymbol{D}^{(t)} z_k^{(t+1)}, \ k = 1, ..., n_s \tag{15}$$

Where

$\boldsymbol{x}_k^{t+1}$ represents the offspring created, t is the generation number, n_s is the population size where generally $n_s = 4 + 3(ln(n))$. U is the number of best individuals that will be recombined (normally $n_s/2$). $\boldsymbol{B}^{(t)}$ represents an orthogonal $n_x \times n_x$ matrix and $\boldsymbol{D}^{(t)}$ a diagonal $n_x \times n_x$ matrix obtained from the eigen decomposition of $\boldsymbol{C}^{(t)}$, $\boldsymbol{C}^{(t)} = \boldsymbol{B}^{(t)} \boldsymbol{D}^{(t)} (\boldsymbol{B}^{(t)} \boldsymbol{D}^{(t)})^T$. The covariance matrix $\boldsymbol{C}^{(t)}$ is symmetric positive definite, and its default initial value is $\boldsymbol{I}$. $\boldsymbol{x}_w^{(t)} = \Sigma_{i=1}^{\mu} \boldsymbol{w}_i \boldsymbol{x}_{i:n_s}^{(t)}$ is the weighted mean of the μ best individuals at the generation and $\boldsymbol{x}_w^{(t)}$ denotes the i^{th} best out of the n_s individuals ranked by the function value.

Simple modifications can be made to the CMA-ES to achieve linear complexity; these include:

- Obtaining the weighted mean of the best individual values for generation g; and
- The recombination of weights (w_i) is positive and if added together are equal to one. To favour the best ranked (μ) individuals,

$$w_i = \frac{ln(\mu + 1) - ln(i)}{\sum_{j=\{1,...,\mu\}} (ln(\mu + 1) - ln(j))}. \tag{16}$$

More of these modifications can be seen in Ros and Hansen (2008). The CMA-ES then uses a global step function implemented as an evolution path. The initial step size $(\sigma^{(0)})$ is a problem dependent parameter and the initial evolution path $P\sigma^{(0)} = 0$. After the initialisation phase the evolution path and step size are calculated as:

$$P\sigma^{(t+1)} = (1 - C_\sigma) P_\sigma^{(t)} + \sqrt{C_\sigma (2 - C_\sigma)} \times \sqrt{\mu_{eff}} B^{(t)} Z_w^{(t+1)} \tag{17}$$

$$\sigma^{(t+1)} = \sigma^{(t)} exp(\frac{c_\sigma}{d_\sigma} (\frac{|p_\sigma^{(t+1)}|}{E(|NM(0, I)|)} - 1)) \tag{18}$$

Where $NM(0, \boldsymbol{M})$ denotes the independent realisations of the multivariate normal distribution, with a covariance matrix $\boldsymbol{M}$. The random vectors $\boldsymbol{z}_k^{(t+1)}$ are $N(0, I)$ distributed and for $\boldsymbol{x}_k^{(t)}$, we can compute their weighted mean: $z_w^{(t+1)} = \Sigma_{i=1}^{\mu} w_i z_{i:n_s}^{(t)}$. $\sigma^{(t)} \in R$ is the step size. $c_\sigma \in [0, 1]$ is the time constant for the adaption of the step size $\sigma^{(t+1)}$. μ_{eff} denotes the variance-effective selection mass and $d_\sigma > 0$ is a damping factor with a default value of $1 + 2 \times max(0, \sqrt{\frac{\mu_{eff} - 1}{n_x + 1}} - 1) + c_\sigma$.

The initial step size is a problem-dependent parameter. The process of sampling individuals and updating of internal strategy parameters is iterated until the desired stopping criterion is reached.

2.1.5 Particle swarm optimisation

Particle swarm optimisation (PSO) has roots that tie it to artificial life; the flocking of birds, fish schooling and swarming theory in general (Eberhart and Kennedy, 1995a). PSO uses basic mathematical operators and requires minimum computational capacity relative to other metaheuristics. The PSO performed well on test functions when compared with a GA when first developed (Eberhart and Kennedy, 1995a). PSO can be used to solve many similar problems to a GA (Eberhart and Kennedy, 1995b). The advantage of the PSO over the GA is the memory of the previous solutions.

In PSO, individuals who fly past optimal solutions will return towards them. The system is initialised with a population of random solutions. It differs from the GA in the way that each potential solution is assigned a velocity, whereafter it is then called a particle, which 'flies' through hyperspace at the assigned velocity. Each particle stores the coordinates in hyperspace of the best solutions it has achieved thus far (the fitness value is also stored). The value of the best-found value of each particle is called the *pbest* value. With regard to the bigger picture there is another 'best' value, the global best value found thus far by any particle, *gbest* (Eberhart and Kennedy, 1995a).

The basic PSO algorithm is discussed in great depth in Eberhart and Kennedy (1995a). Throughout the optimisation process of the PSO the velocity and displacement of each particle is changed, moving the particle to a new position. The magnitude and direction of a particle at time t is shown in Figure 4 (Grobler et al., 2008). A particle's velocity is the result of three vectors. The particle's velocity at time $(t+1)$, the cognitive component (*pbest*), and the social component (*gbest*).

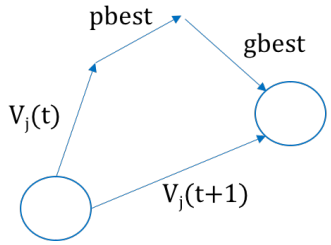

Figure 4: PSO Particle velocity given the three components (Grobler et al., 2008)

.

As discussed earlier the *gbest* model calculates the velocity of a particle i in the dimension j at time $t+1$ as:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[\widehat{x}_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[x_j^*(t) - x_{ij}(t)] \tag{19}$$

Where $v_{ij}(t)$ is the velocity of particle i in dimension j at time t; $c_{1/2}$ are the cognitive and social acceleration constants respectively; and $\widehat{x}_{ij}/x_{ij}(t)$ are the *pbest* and position of a particle i in dimension j at time t respectively. $x_j^*(t)$ represents the global best position (*gbest*) in dimension j and w is the inertia weight. $r_{1j}(t)$ is a random sample from uniform random distribution $U(0,1)$ and $r_{2j}(t)$ is a random sample from uniform random distribution $U(0,1)$.

The displacement of a particle i at time t is simply derived from $v_{ij}(t+1)$ as:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{20}$$

It is because of the simultaneous movement of particles toward their previous best solution (*pbest*) and the best solution found by the entire swarm (*gbest*) that the particles converge to one or more good solutions in the search space. Grobler et al. (2008) went on to provide the pseudocode for the basic PSO, shown in Algorithm 8.

Algorithm 8: Basic PSO algorithm

Initialise an n_x-dimensional swarm of n_s particles
$t = 1$
while $t < I_{max}$ **do**
  for *each particle* i **do**
    if $f(\boldsymbol{x}_i(t)) \leq f(\widehat{\boldsymbol{x}}_i(t))$ **then**
      $\widehat{\boldsymbol{x}}_i(t) = \boldsymbol{x}_i(t)$
    if $f(\widehat{\boldsymbol{x}}_i) < f(\boldsymbol{x}^*)$ **then**
      $\boldsymbol{x}^* = \widehat{\boldsymbol{x}}_i$
  end
  for *each particle* i **do**
    $v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[\widehat{x}_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[x_j^*(t) - x_{ij}(t)]$
    $x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1)$
  end
  $t = t + 1$
end

The basic algorithm for the PSO is not guaranteed to converge. Van den Bergh and Engelbrecht (2002) showed that non-convergence will happen if any particle reaches the position where:

$$\widehat{x} = x(t) = x^* \tag{21}$$

Given the obstacle of non-convergence, the guaranteed convergence particle swarm optimisation (GCPSO) algorithm introduced by Van den Bergh and Engelbrecht (2002) has been shown to address this problem effectively. The GCPSO requires a different velocity and displacement update shown below in Equations (22) and (23).

$$v_{rj}(t+1) = -x_{rj}(t) + x_j^*(t) + wv_{rj}(t) + p(t)(1 - 2r_j(t)) \tag{22}$$

$$x_{rj}(t+1) = x_j^*(t) + wv_{rj}(t) + p(t)(1 - r_j(t)) \tag{23}$$

The equations listed above are applied to the global best particle where $p(t)$ is a time-dependent scaling factor and $r_j(t)$ is sampled from a uniform random distribution, $U(0, 1)$. All the other particles are still updated using Equations (19) and (20) (Grobler et al., 2008). The pseudocode for the GCPSO can be seen in Algorithm 9.

Algorithm 9: GCPSO algorithm

Initialise an n_x-dimensional swarm of n_s particles
$t = 1, p(t) = 1, \zeta = 0, \eta = 0$
while $t < I_{max}$ **do**
  for *each particle* i **do**
    if $f(x_i(t)) \leq f(\widehat{x}_i)$ **then**
      $\widehat{x}_i = x_i(t)$
    if $f(\widehat{x}_i) < f(x^*)$ **then**
      $\eta = 0$
      $x^* = \widehat{x}_i$
    else if then
      $\eta = \eta + 1$
      $\zeta = 0$
  end
  for *each particle* $i | i \neq \tau$ **do**
    $v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[\widehat{x}_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[x^*_j(t) - x_{ij}(t)]$
    $x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1)$
    $v_{rj}(t+1) = -x_{rj}(t) + x^*_j(t) + wv_{rj}(t) + p(t)(1 - 2r_j(t))$
    $x_{rj}(t+1) = x^*_j(t) + wv_{rj}(t) + p(t)(1 - r_j(t))$
  end
  $t = t + 1$
end

2.1.6 Summary

This section presented suitable solution strategies for the part picking and allocation problem. Each of these search methodologies was described in detail. The four metaheuristics chosen as suitable solution strategies include the CMA-ES, GA, GCPSO and the SaNSDE. The next section discusses the multi-objective optimisation algorithms used to solved the multi-objective part picking and allocation problem.

2.2 Multi-objective optimisation

Multi-objective optimisation, in general, has been receiving increasing attention over the past few years. This section focuses on the multi-objective optimisation problem.

A multi-objective optimisation (MOO) problem can be formally defined as follows:

$$\text{Minimise } \boldsymbol{f}(\boldsymbol{x}) \tag{24}$$

$$\text{subject to } \boldsymbol{\zeta_q}(\boldsymbol{x}) \leq 0, \quad q = 1, ..., n_\zeta \tag{25}$$

$$\boldsymbol{\varrho_q}(\boldsymbol{x}) = 0, \quad q = n_\zeta + 1, ..., n_\zeta + n_\varrho \tag{26}$$

$$\boldsymbol{x} \in [\boldsymbol{x_{min}}, \boldsymbol{x_{max}}]^{n_x} \tag{27}$$

where $\boldsymbol{f}(\boldsymbol{x})$ denotes the vector of the objective function to be minimised, $\boldsymbol{\zeta_q}$ and $\boldsymbol{\varrho_q}$ are the inequality and equality constraints respectively, and $\boldsymbol{x} \in [\boldsymbol{x}_{min}, \boldsymbol{x}_{max}]^{n_x}$ represent the boundary constraints. A solution to a MOO problem can thus be defined as a vector $\boldsymbol{x}$ that satisfies the constraints and optimises the vector function $\boldsymbol{f}(\boldsymbol{x})$ (Zitzler et al., 1999). A large number of approaches have already been documented in literature to optimise conflicting objectives.

Fortunately, Werner (2006) provided a generic classification which differentiates between different MOO approaches depending on the stage of the optimisation process at which the decision makers' preferences are incorporated as seen in Figure 5. The first of these strategies, known as *priori* MOO techniques, incorporates the decision makers' preferences at the start of the optimisation process. Examples of preference information include relative importance or targets for each objective function. This information is then incorporated into the optimisation process. Since most *a priori* techniques transform the multi-objective problem into a single objective problem, only one solution to the scheduling problem can be obtained at a time. One of the most popular examples of this strategy includes weighted aggregation where all objective functions are combined into a linear combination of criteria (Rardin and Rardin, 1998).

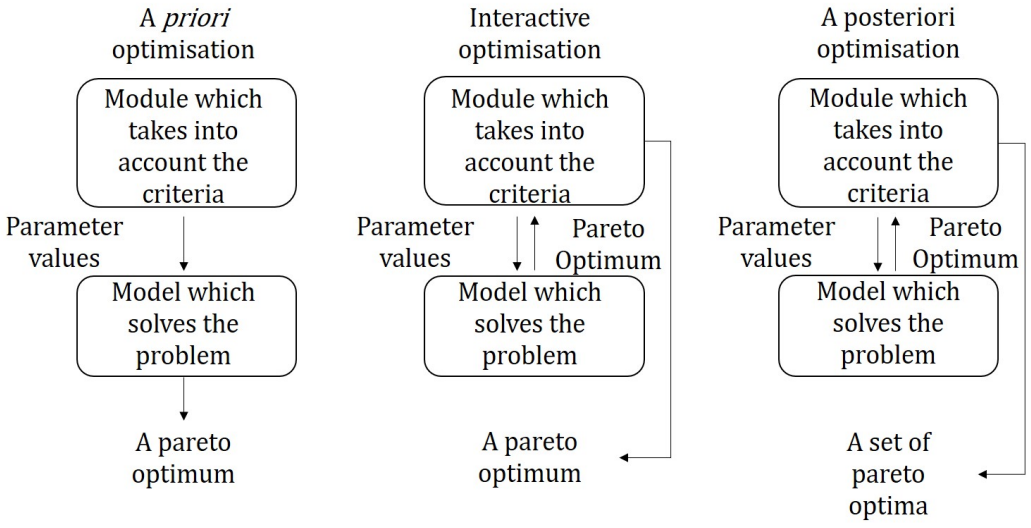

Figure 5: Three main strategies for addressing multiple conflicting objectives (Rardin and Rardin, 1998)

Interactive MOO methods involve the decision-makers throughout the optimisation process. Decision-makers who specify weight changes at intermittent stages of the optimisation process are an example of an interactive approach. This strategy is useful in steering the algorithm to more desirable regions of the objective space. This research is more concerned with a *priori* versus a *posteriori* optimisation; however, a more detailed study of interactive methods can be found in Vanderpooten (1990). No user preferences are taken into account before or during the optimisation process in *a posteriori* MOO. The focus is on providing the decision-makers with as diverse a set of solutions as possible to facilitate the selection of the most suitable solution from the set (Werner, 2006). The purpose of *a posteriori* MOO can thus be summarised as finding a set of trade-off solutions referred to as the Pareto optimal set, $\boldsymbol{PF}$.

Evolutionary multi-objective optimisation (EMO) algorithms are a branch of evolutionary algorithms. EMO algorithms were investigated by Wong et al. (2010) who provided a short survey. A more detailed review of EMO algorithms can be found in Shir et al. (2010), Stoean et al. (2010), and Coello et al. (2007).

The following section introduces the metaheuristics that are used to solve the multi-objective part sequencing and allocation problem in the algorithm framework. The next sections discuss the multi-objective optimisation algorithms, which includes the non-dominated sorting genetic algorithm III (NSGA-III) in Section 2.2.1, the multi-objective evolutionary algorithm based on decomposition (MOEAD) in Section 2.2.2, the multiple objective particle swarm optimisation (MOPSO) are discussed in Section 2.2.3, and the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) in Section 2.2.4.

2.2.1 The non-dominated sorting genetic algorithm III

The structure of the NSGA-III is very similar to the original NSGA-II algorithm (Deb et al., 2002). Significant changes have been made with regard to its selection operator. In contrast to the NSGA-II, the maintenance of the population diversity is improved by creating and adaptively updating the number of well-spread reference points (Deb and Jain, 2013). As described in Deb and Jain (2013), consider the t^{th} generation of the NSGA-II algorithm. Let the parent population at the respective iteration be $\boldsymbol{P_t}$, and its size n_s. The offspring of $\boldsymbol{P_t}$ is then $\boldsymbol{Q_t}$ with n_s number of members. The first step will be to identify the best performing n_s members from both the parent ($\boldsymbol{P_t}$) and the offspring ($\boldsymbol{Q_t}$) given $\boldsymbol{R_t} = \boldsymbol{P_t} \cup \boldsymbol{Q_t}$ (size $2n_s$). In order to identify the best performing members the combined population $\boldsymbol{R_t}$ is sorted based on non-domination order $(F_1, F_2, ..., F_N)$. Each non-dominated level is then used to develop a new population $\boldsymbol{S_t}$. If the last level is denoted as the l^{th} level, all solutions from $(l+1)$ onward are rejected from the combined population $\boldsymbol{R_t}$. In NSGA-II this is achieved through a niche preservation operator that computes the crowding distance for every last member in a level. The crowding distance is calculated as the summation of an objective-wise normalised distance between two neighbouring solutions. The solutions with the higher crowding distance values are chosen. In the NSGA-III, the crowding distance operator is changed using the following approaches (Deb and Jain, 2013):

- Classification of population into non-dominated levels,
- Determination of reference points on a hyperplane,
- Adaptive normalisation of population members,
- Association operation,
- Niche-preservation operation.

The above-mentioned operators are described in more detail in Deb and Jain (2013).

Jain and Deb (2013) discussed the many objective NSGA-II or NSGA-III in detail. For the initialisation phase the parent population $\boldsymbol{P_t}$ (of population size n_s) is randomly initialised, thereafter the binary tournament selection, cross-over and mutation operators create an offspring population $\boldsymbol{Q_t}$. After that the two populations are combined and sorted according to their domination level (Jain and Deb, 2013). The best n_s members then form the next generation. Unlike NSGA-II, NSGA-III starts with a set of reference points $\boldsymbol{Z^r}$. After the domination sorting all front members and the last front ($\boldsymbol{F_l}$) which could not be completely accepted are saved in a set $\boldsymbol{S_t}$. Members that are part of $\boldsymbol{S_t}$ and $\boldsymbol{F_l}$ are automatically part of the next generation; however, the remaining members are selected from $\boldsymbol{F_l}$ in a way that the desired diversity is maintained in the population.

The NSGA-II used the crowding distance measure to select a well-distributed set of points. The NSGA-III used the supplied reference points $\boldsymbol{Z^r}$ to select the remaining members. In order to achieve identical ranges, the objective values and the reference points need to be normalised. Thereafter the orthogonal distance between a member in $\boldsymbol{S_t}$ and each of the reference lines (joining an ideal point and a reference point) is calculated. The member will then be associated with the reference point having the smallest orthogonal distance. After that point has been determined, the niche count $\boldsymbol{\rho}$ for each reference point is defined as the number of members in $\boldsymbol{S_t}$ and $\boldsymbol{F_l}$ that are associated with the reference point. The reference that has the lowest niche count is identified and the member that is before $\boldsymbol{F_l}$ is then associated with the identified reference point and is included in the final population.

The niche count is increased by one and the process is repeated to fill up the population $\boldsymbol{P}_{t+1}$. Algorithm 10 presents the algorithm used in Jain and Deb (2013) to generate generation t of the NSGA-III.

Algorithm 10: NSGA-III, the process to create generation t (Jain and Deb, 2013)

Input: $\boldsymbol{H}$ reference points $\boldsymbol{Z^r}$, parent population $\boldsymbol{P_t}$
Output: $\boldsymbol{P}_{t+1}$
$\boldsymbol{S_t} = \emptyset$, $i = 1$
$\boldsymbol{Q_t}$ = Recombination + Mutation($\boldsymbol{P_t}$)
$\boldsymbol{R_t} = \boldsymbol{P_t} \cup \boldsymbol{Q_t}$
$(F_1, F_2, ...)$ =Non-dominated sort ($\boldsymbol{R_t}$) **while** $|\boldsymbol{S_t}| < n_s$ **do**
    $\boldsymbol{S_t} = \boldsymbol{S_t} \cup F_i$
    $i = i + 1$
end
Last front to be included: $F_l = F_i$
if $|\boldsymbol{S_t}| = N$ **then**
    $\boldsymbol{P}_{t+1} = \boldsymbol{S_t}$, break
else
$\boldsymbol{P}_{t+1} = \cup_{j=1}^{l-1} F_j$
Points to be chosen from $F_l : \boldsymbol{K} = N - |\boldsymbol{P}_{t+1}|$
Normalise objectives
Associate each members of the set $\boldsymbol{S_t}$ with a reference point: $[\pi(s), d(s)]$ = Associate $(\boldsymbol{S_t}, Z^r)\%\pi(s)$: closest reference point, d: distance between s and $\pi(s)$
Compute niche count of reference point $j \in \boldsymbol{Z^r} : \boldsymbol{\rho_j} = \sum_{s \in \boldsymbol{S_t}/F_l}((\pi(s) = j)(1:0)$
Choose $\boldsymbol{K}$ members one at a time from F_l to construct $\boldsymbol{P}_{t+1}$: Niching $(\boldsymbol{K}, \boldsymbol{\rho_j}, \pi, d, \boldsymbol{Z^r}, F_l, \boldsymbol{P}_{t+1})$

2.2.2 The multi-objective evolutionary algorithm based on decomposition

Decomposition is a basic strategy in multi-objective optimisation. Although it has been used in multi-objective optimisation it has not been used widely in evolutionary multi-objective optimisation (Zhang and Li, 2007).

The MOEAD algorithm decomposes the multi-objective problem into a number of scalar optimisation problems. After the decomposition, an evolutionary algorithm is employed for solving these subproblems simultaneously. All the individual solutions in the population are associated with a subproblem. A neighbourhood relationship among all subproblems is defined based on the distance of their weight vectors. In summary, the MOEAD optimises a number of subproblems and then uses the information from the neighbouring subproblems to move to near-optimal solutions.

In the following description of the MOEAD the Tchebycheff approach was employed (Zhang and Li, 2007). Let $\boldsymbol{\lambda^1}, ..., \boldsymbol{\lambda^E}$ be a set of even spread weight vectors and $\boldsymbol{z^*}$ be the reference point. The objective function of the j^{th} subproblem is (Zhang and Li, 2007):

$$\boldsymbol{g^{te}(x|\lambda^j, z^*)} = max(l \leq i \leq m)\{\boldsymbol{\lambda_i^j} f_i(\boldsymbol{x}) - \boldsymbol{z^*}|\} \tag{28}$$

where $\boldsymbol{\lambda^j} = (\boldsymbol{\lambda_1^j}, ..., \boldsymbol{\lambda_m^j})^T)$. The MOEAD algorithm minimises all these E objective functions simultaneously. The optimal solution of $\boldsymbol{g^{te}(x|\lambda^i, z^*)}$ should be close to that of $\boldsymbol{g^{te}(x|\lambda^j, z^*)}$ if $\boldsymbol{\lambda^i}$ and $\boldsymbol{\lambda^j}$ are close to each other. Information about these $\boldsymbol{g^{te}}$'s with weight vectors close to $\boldsymbol{\lambda^i}$ benefits the optimising of $\boldsymbol{g^{te}(x|\lambda^j, z^*)}$.

In the MOEAD algorithm, the Tchebycheff approach evaluates Algorithm 11 at each generation t,:

Algorithm 11: MOEAD at each generation t

1 Initialise a population of n_s points $\boldsymbol{x}^1, ..., \boldsymbol{x}^N \in \boldsymbol{\Omega}$, where $\boldsymbol{x}^i$ is the current solution to the i^{th} sub-problem
2 $\boldsymbol{FV}^1, ..., \boldsymbol{FV}^N$, where $\boldsymbol{FV}^i$ is the $\boldsymbol{F}$-value of $\boldsymbol{x}^i$, i.e. $\boldsymbol{FV}^i = \boldsymbol{FV}(\boldsymbol{x}^i)$ for each $\boldsymbol{i} = 1, ..., N$
3 $\boldsymbol{z}(\boldsymbol{z}_1, ..., \boldsymbol{z}_m)^{\boldsymbol{T}}$, where $\boldsymbol{z}_i$ is the best value found so far for objective f_i
4 Create an external population ($\boldsymbol{EP}$), which is used to store non-dominated solutions found during the search for each generation.

2.2.3 The multiple objective particle swarm optimisation

Extending the particle swarm optimisation with evolutionary algorithms to multi-objective optimisation can be done with the use of a Pareto ranking scheme (Golberg, 1989). The repository of best-found solutions can be used to store non-dominated solutions generated (similar to the notion of elitism as used in evolutionary multi-objective optimisation) (Coello and Lechuga, 2002). The strategy is to use global attraction mechanisms combined with a repository of previously found non-dominated solutions, which would enhance convergence towards globally non-dominated solutions. The algorithm keeps data in a global repository in which every particle will store its flight data. Additionally, the updates in the global repository are performed considering a geographically-based system defined by the objective function. This technique is inspired by the Pareto archive evolution strategy (PAES) (Knowles and Corne, 2000). The algorithm for the multi-objective particle swarm optimisation algorithm as described by Coello and Lechuga (2002) is:

Algorithm 12: MOPSO Algorithm

```
Initialise the population
Initialise the speed of each particle
Evaluate each particle in the population
Store the geographical positions of each non-dominated solution in the repository
Generate hyper-cubes of the search space explored
Initialise the memory of each particle (This memory is also stored in the repository)
while maximum number of iteration has not been reached do
    Compute the new speed of each particle
    Compute the new position of each particle
    Maintain the particles within the search space
    Evaluate each particle in the population
    Update the repository
    Update the particle's memory if the current position is better than the position
     stored
    Increment the loop counter
end
```

2.2.4 The multi-objective covariance matrix adaptation evolution strategy

The multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) algorithm is an evolutionary strategy and is designed to learn the relationships and dependencies between a given set of decision variables by adapting a covariance matrix. The performance of the MO-CMA-ES was tested on various techniques and multi-objective test functions (Igel et al., 2007). The standard CMA-ES relies on the non-elitist $(\boldsymbol{\mu}, \boldsymbol{\lambda})$ selection. The $\boldsymbol{\mu}$ will then form the next parent population and all older populations are discarded. In order to maximise the number of different strategy parameter sets, given a fixed population size, the number of offspring per parent must be as small as possible (Igel et al., 2007). First, a single objective elitist $(1 + \boldsymbol{\lambda})$-CMA-ES with $(1 + \boldsymbol{\lambda})$ selection was developed, where $\boldsymbol{\lambda}$ can be as small as 1. In the single objective $(1 + \boldsymbol{\lambda})$-CMA-ES the parent population consists of a single individual generating $\boldsymbol{\lambda}$ offspring and the best individual from the offspring and parent population becomes the parent of the next generation. This means that the $(1+\boldsymbol{\lambda})$-CMA-ES inherits all in-variance properties from the original CMA-ES. The in-variance properties are then integrated into the MOO framework by considering a population of $(1 + \boldsymbol{\lambda})$ evolution strategies.

In the multi-objective CMA-ES $\boldsymbol{\lambda}_{MO} \times (1 + \boldsymbol{\lambda})$-MO-CMA-ES is a population of $\boldsymbol{\lambda}_{MO}$ elitist $(1 + \boldsymbol{\lambda})$-CMA-ES. The i^{th} individual in iteration t is denoted as

$$\boldsymbol{x}_i^{(t)} = [\boldsymbol{x}_0^{(t)}, \bar{\boldsymbol{p}}_{succ,i}^{(t)}, \sigma_i^{(t)}, \boldsymbol{p}_{c,i}^{(t)}, \boldsymbol{C}_i^{(t)}]$$

where $\boldsymbol{x_i}$ is the initial candidate solution, $\boldsymbol{p}_{succ,i}^{(t)}$ is the success probability and $\boldsymbol{\sigma(t)}$ denotes the global step size. For simplicity Igel et al. (2007) considered the standard case $\boldsymbol{\lambda} = 1$. In each iteration (t) each of the parents $\boldsymbol{\lambda}_{MO}$, generates $\boldsymbol{\lambda} = 1$ offspring. The parents and their offspring form the set $\boldsymbol{Q^{(t)}}$. The step size of a parent and its offspring are updated depending on whether the mutations were successful. The mutations are classified as successful when the offspring is better than the parent according to the relation $\prec_{\boldsymbol{Q}^{(t)}}$, where $\prec$ refers to the precedence. The algorithm as described in Igel et al. (2007) is shown in Algorithm 13

Let us denote:

$$\boldsymbol{\lambda}^{(t+1)}_{succ,\boldsymbol{Q^{(t)}},i} = \begin{cases} 1 & \text{if } \alpha_i^{'(t+1)} \prec_{\boldsymbol{Q}^{(t)}} \alpha_k^{(t)} \text{ is the number of successful offspring} \\ & \text{from parent } \alpha_i^{(t)} \text{ for } \boldsymbol{\lambda} = 1 \text{ and } \boldsymbol{Q}^{(t)}_{\prec:i} \text{ is the } i^{\text{th best}} \\ & \text{offspring in } \boldsymbol{Q}^{(t)} \text{ w.r.t. } \prec_{\boldsymbol{Q}^{(t)}} \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 13: MO-CMA-ES

$t = 0$, initialise $\boldsymbol{x}_i^{(t)}$ for $i = 1, ..., \boldsymbol{\lambda}_{MO}$
while *maximum number of iteration has not been reached* **do**
  for $i = 1, ..., \boldsymbol{\lambda}_{MO}$ **do**
    $\boldsymbol{x}_i^{'(t+1)} \leftarrow \boldsymbol{x}_i^{(t)}$
    $\boldsymbol{x}_i^{'(t+1)} \sim N(\boldsymbol{x}_i^{(t)}, \boldsymbol{\sigma}_i^{(t)^2} \boldsymbol{C}_i^{(t)})$
  end
  $\boldsymbol{Q}^{(t)} = \{\boldsymbol{x}_i^{'(t+1)}, \boldsymbol{x}_i^{(t)} | 1 \leq i \leq \boldsymbol{\lambda}_{MO}\}$ **for** $i = 1, ..., \boldsymbol{\lambda}_{MO}$ **do**
    update step size $(\boldsymbol{x}_i^{(t)}, \boldsymbol{\lambda}^{(t+1)}_{succ,\boldsymbol{Q}^t,i})$
    update step size $(\boldsymbol{x}_i^{'(t+1)}, \boldsymbol{\lambda}^{(t+1)}_{succ,\boldsymbol{Q}^t,k})$
    update step covariance $(\boldsymbol{x}_i^{'(t+1)}, \frac{\boldsymbol{x}_i^{'(t+1)} - \boldsymbol{x}_i^{(t)}}{\boldsymbol{\sigma}_i^{(t)}})$
  end
  for $i = 1, ..., \boldsymbol{\lambda}_{MO}$ **do**
    $\boldsymbol{x}_i^{(t+1)} \leftarrow \boldsymbol{Q}^{(t)}_{\prec:i}$
  end
  $t \leftarrow t + 1$
end

It is notable that a MO-CMA-ES has never been tested on this specific problem. The contribution of this research includes the incorporation of collision avoidance in the robot routing multi-objective problem while using metaheuristics as solution strategies, thus to the best of the author's knowledge, this research is the first where the MO-CMA-ES is benchmarked against the MOPSO, NSGA-III, and MOEAD, given that all collisions must be avoided.

2.2.5 Multi-objective performance metrics

The following popular multi-objective performance metrics were considered.:

1. Hypervolume (HV) (Zitzler and Thiele, 1999): The hypervolume metric calculates the area of the objective space created by the Pareto front, from a given reference point.

2. Number of Pareto solutions (NPS) (Hassan-Pour et al., 2009): The NPS calculates the total number of Pareto optimal solutions found for each evaluation run.

3. Spacing metric (SM) (Deb et al., 2000): The spacing metric calculates the uniformity of the spacing of the points on the Pareto front. The SM is calculated as:

$$SM = [\frac{1}{n-1} \sum_{i=1}^{m} (\overline{\boldsymbol{d}} - \boldsymbol{d}_i)^2)]^{0.5} \tag{29}$$

$$\boldsymbol{d}_i = \sqrt{(q_2 - q_1)^2 + (y_2 - y_1)^2)} \tag{30}$$

Where $\overline{\boldsymbol{d}}$ is the mean value of all $\boldsymbol{d}_i$, where $\boldsymbol{d}_i$ is the distance between two succeeding non-dominated Pareto solutions and $p(q_1, q_2), p(y_1, y_2)$ represents the coordinates of the Pareto solution.

2.3 Summary

Section 2.1 discussed the single objective robot routing literature as well as the four single objective metaheuristics chosen as suitable solution strategies; namely, the GA, DE, CMA-ES, and PSO. Section 2.2 presented suitable solution strategies for the multi-objective part picking and allocation problem. Each of these search methodologies was described in detail. The four algorithms chosen as suitable solution strategies include the MO-CMA-ES, NSGA-III, MOPSO and the MOEAD.

Chapter 3

Automated warehousing literature review

Warehouses are typically simple buildings used for storing products and goods. Warehouses allow a supply chain to keep buffer stock at a predetermined position in the supply chain. They can be used in different supply chains for various reasons including; safety stock, long lead time items, space constraints, or simply a business requirement. Warehouses usually have dedicated loading and unloading areas for transportation vehicles. The aim of a warehouse is to handle these vehicles and associated picking and binning operations as efficiently as possible. Section 3.1 discusses all activities related to warehousing and possible warehouse layouts. Section 3.2 discusses special cases of the travelling salesman problem that are compared to the robot routing and allocation problem. Section 3.3 introduces the automated guided vehicle and automated guided vehicle systems. Section 3.4 discusses single objective robot path planning and task allocation literature and Section 3.5 discusses the multi-objective robot path planning and task allocation literature. Section 3.6 summarises the chapter.

3.1 Warehousing activities

Warehousing activities form an important part of a business's supply chain. From a financial perspective, warehousing costs amount to approximately 20% of the total logistics costs (Baker and Halim, 2007). Order picking and binning have been identified as the most time-consuming general functions in a warehouse (Roodbergen and Koster, 2001). Although many companies minimise the need for warehouse space by creating just-in-time (JIT) systems or by creating a synchronised supply system, there is still a need for inventory warehouses. Warehouses can be used for storing raw materials, work-in-progress parts, and finished goods products (Baker and Canessa, 2009).

Considering the relationship between warehouse costing and customer satisfaction, companies often explore new possibilities to improve their warehouse efficiency. Automation equipment like conveyor systems and automated picking and binning equipment is frequently implemented to improve the efficiency of a warehouse (Baker, 2004).

Implementation of automated warehousing equipment and systems is often the step where businesses encounter difficulties (Wyland, 2008). Before a business can implement any warehouse automation processes the following topics have to be well defined: the strategic action, the organisation's capabilities and the technologies available for automation. Wyland (2008) created a list of actions required to obtain the best results when optimising a warehouse:

- Create a constant flow and visualisation methods;
- Connect the workforce to the business requirements; and
- Adopt technological changes.

Moeller (2011) discovered that automatic order picking and binning is mostly advantageous for product ranges that are fixed and where no flexibility or customisation occurs. The products must be standard and not change frequently. Andelković and Radosavljević (2018) investigated the improvement opportunities of using a warehouse management system. Andelković and Radosavljević (2018) found that there are a number of factors influencing the service quality level, such as; order delivery time, order integrity, and order accuracy. The quicker the order can be completed and delivered to the customer, the more satisfied the customer will be and the company's cost to serve the customer will decrease. The optimisation of the order picking process includes the optimisation of the following phases during the order picking process (Broulias et al., 2005):

- Travelling time required;
- Travelling distance to be travelled by the vehicle;
- Picking or binning time; and
- Return time to transport the products to the dispatch point.

The initial layout design of a warehouse defines the number of aisles, number of racks, width of aisles, width of racks, number of cross aisles, and the height of the roof. The number of layout options for a warehouse is endless, though certain options might be more suitable or more efficient than other options. The layout options for a warehouse have a major impact on the efficiency of order picking and robot travelling times. Layout design parameters such as the number of cross aisles and the shape of the warehouse have a more than 60% effect on total travelling distances in the warehouse (Karásek, 2013). De Koster et al. (2007a) and Heragu (2008) listed a few characteristics to be considered when designing a new warehouse, which include: the number of blocks in the warehouse; the length of the warehouse; the width of the warehouse; number of picking aisles; the number and shape of the cross aisles; the number of rack levels to be used for storage; the position of the entry gate and the number of entry and exit gates.

Given all of the characteristics to consider, Karásek (2013) investigated the traditional layouts as well as a number of newly adapted designs such as the flying V layout, fishbone layout, and inverted V layout. Gue and Meller (2009) introduced new flying V and fishbone layouts with a reduction of 10% to 20% in travelling distance. Gue et al. (2012) further improved the flying V and inverted V layouts, and the improved designs decreased the travelling distance by a further 3%. The illustrations of the layout examples discussed above are shown in Figure 6.

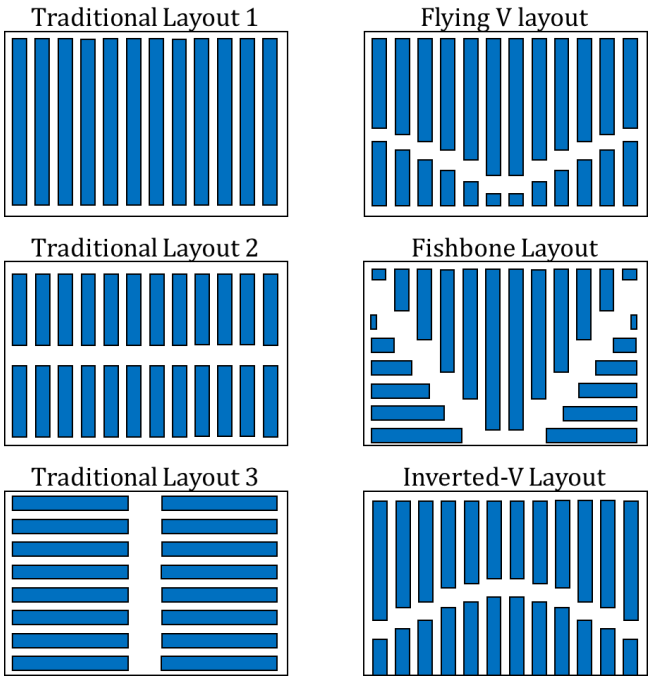

Figure 6: Warehouse configuration examples

A computational system used to solve the warehouse aisle design problem was investigated by Ozden (2017), who stated that in order to solve the order picking warehouse layout problem, a few problems needed to be solved simultaneously:

1. Layout design problem;
2. Product allocation;
3. Picker routing;
4. Location allocation;
5. Batching of parts;
6. Centralised vs decentralised;
7. Collision avoidance; and
8. Sequencing of orders.

Li et al. (2017) focused on the batching and picking routing for online retailers and fast-moving consumer goods. An algorithm was developed for the batching of products and the picking routes. The similarity coefficient algorithm was the most effective method to batch the orders. After the batching process, the routing was optimised using ant colony optimisation with a local search. The algorithm was tested on actual sales data and the results were positive with a reduction in manpower spent on picking and smaller batch sizes to be picked. For the batching and picking route optimisation, Li et al. (2017) used integer programming with the objective of minimising the total travelling distance rather than travelling time.

3.2 The travelling salesman problem and Steiner travelling salesman problem

The travelling salesman (TSP) problem is important for this dissertation because the robot routing and allocation problem has been described as a special case of the TSP. The TSP is a problem frequently found in operations research. It can easily be expressed as a graph describing the servicing of locations made up of a set of nodes, by minimising an objective value. A typical example is the route a postman has to travel to deliver his letters, using the shortest route. The methodology used in Kim et al. (2003) included a simple sorting based heuristic and an efficient clustering-based algorithm. Han et al. (1987) also investigated the block sequencing approach and concluded that the problem of sequencing a number of products for picking and binning was similar to the TSP.

Burkard et al. (1998) describe the TSP as a salesman that needs to visit each city on a given list once, starting at the salesman's house, visiting all the cities and then ending the route at his house again. He can choose in which order he visits the cities but all cities must be visited. He certainly wants to minimise the distance travelled. As the salesman attempts to find the shortest route, he finds himself solving the travelling salesman problem. Dantzig et al. (1954) formulated the TSP by labelling the cities $1, ..., b$ and defined the following:

$$u_{de} = \left\{ \begin{array}{l} 1 \text{ if the path goes from city } d \text{ to city } e \\ 0 \text{ otherwise} \end{array} \right\} \tag{31}$$

Take $distance_{de} > 0$ to be the distance from a city d to city e. The objective for the model is:

$$Min \sum_{d=1}^{b} \sum_{e \neq d, d=1}^{b} distance_{de} \times u_{de} \tag{32}$$

Subject to,

$$\sum_{d=1, d \neq e}^{b} u_{de} = 1 \qquad e = 1, ..., b, \tag{33}$$

$$\sum_{e=1, e \neq d}^{b} u_{de} = 1 \qquad d = 1, ..., b, \tag{34}$$

$$\sum_{d \in \boldsymbol{H}} \sum_{e \neq d, e \in \boldsymbol{H}} u_{de} \leq |\boldsymbol{H}| - 1 \qquad \forall \boldsymbol{H} \not\subseteq 1, ..., b, |\boldsymbol{H}| \geq 2 \tag{35}$$

The constraint shown in Equation (35) ensures no proper subset $\boldsymbol{H}$ can form a sub-tour. The TSP has many applications in routing and production schedules with job-dependent set-up times (Burkard et al., 1998). Because of the difficulty of solving a TSP problem, a need grew for good suboptimal solution methods. Burkard et al. (1998) studied the research on solvable special cases for the TSP by Gilmore (1985) and turned their focus to collecting and summarising new results obtained since the release of Gilmore (1985) to complement that work.

The travelling salesman problem has been well researched in the past by numerous researchers. Karásek (2013) used a self-developed algorithm based on the basic TSP to solve an order picking problem. Roodbergen and Koster (2001) used the branch and bound procedure for the travelling salesman problem to obtain the shortest order picking routes. The branch and bound algorithm was also used as the benchmark in the performance analysis of Roodbergen and Koster (2001).

Molnar and Lipovszki (2005) used a special case of the travelling salesman problem along with a genetic algorithm to solve the problem of routing and scheduling of order vehicles in a warehouse. Wu et al. (2002) investigated the different heuristic strategies on a multi-depot location routing problem. Burkard et al. (1998) solved a special case of the TSP by developing a combinatorial optimisation algorithm.

The graphical TSP is a relaxation of the TSP which allows the travelling agent to visit each city on multiple occasions and use an edge more than once. For the specialised case of order picking, Burkard et al. (1998) considered the Steiner TSP, a related problem.

The Steiner TSP problem constructs a graph $\boldsymbol{G} = (\boldsymbol{V} : \boldsymbol{E})$ and $\boldsymbol{P} \subseteq \boldsymbol{V}$, where $\boldsymbol{V}$ is the vertices set and $\boldsymbol{E}$ is the edge set. It is the elements of $\boldsymbol{V}$ and $\boldsymbol{P}$ that are called Steiner points. A closed walk that visits all the vertices of $\boldsymbol{P}$ at least once is called a Steiner tour of $\boldsymbol{G}$. An example of graph $\boldsymbol{G}$ can be seen in Figure 7.

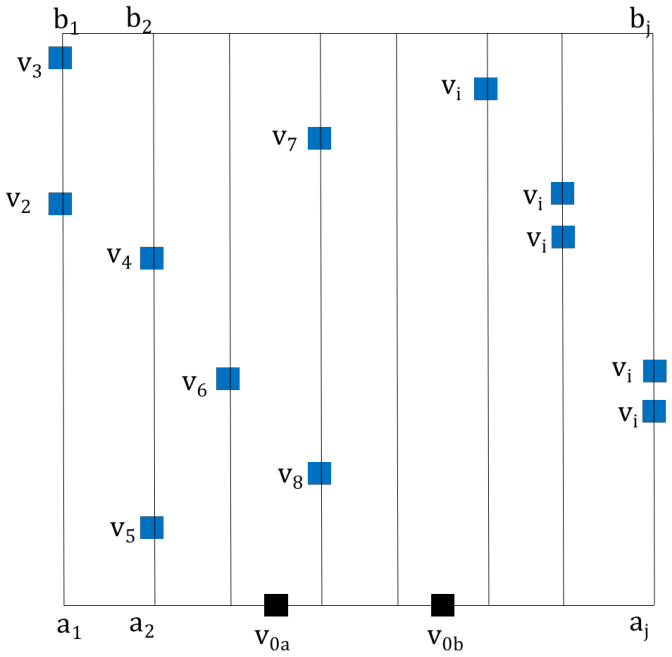

Figure 7: Warehouse configuration graph $\boldsymbol{G}$

The main differences between a TSP and the Steiner tour are:

- In a Steiner tour the Steiner points do not have to be visited, whereas in a TSP all points must be visited; and
- The Steiner tour may visit some vertices on multiple occasions.

It is for the above reasons that the Steiner TSP can be used to formulate the real-world problem of finding a minimum length order picking route. In order to solve the order picking problem efficiently, some assumptions have to be made on the layout configuration. An example of such a layout is the graph of a rectangular warehouse (see Figure 1). The rectangular configuration is the one most frequently encountered in practice.

The points of interest (vertices) in $\boldsymbol{V}$ and $\boldsymbol{P}$, the Steiner points, represent the intersections of aisles. The distance between any two vertices is the length of the shortest path between the two vertices. In order to classify a Steiner TSP as a Graphical TSP, the condition of $\boldsymbol{P} = \boldsymbol{V}$ must be true, which means that all vertices must be visited and some vertices or edges can be used multiple times. Although these rules differ from the classical TSP problem, Fonlupt and Naddef (1992) proved that it is possible to characterise those graphs and find an optimal solution to the relaxation problem of the graphical TSP. Given an $n \times n$ distance matrix $\boldsymbol{C} = (c_{ij})$. All $\boldsymbol{S} \subset \{1, ..., n\} : \boldsymbol{S} \neq \boldsymbol{\emptyset}$ (these graphs are called TSP-perfect graphs), find a cyclic permutation of π of the set $i \in \{1, 2, 3, ..., n\}$ that minimises the function:

$$\sum_{i=1}^{n} \sum_{j>i}^{n} c_{ij} x_{ij} \tag{36}$$

subject to:

$$x_{ij} \geq 0, i, j = 1 : ... : n : i < j \tag{37}$$

and

$$\sum_{i \in \boldsymbol{S}} \sum_{j \notin \boldsymbol{S}: j>i} x_{ij} + \sum_{i \notin \boldsymbol{S}} \sum_{j \in \boldsymbol{S}: j>i} x_{ij} \geq 2 \tag{38}$$

$$x_{ij} \in \{0, 1\} \forall i, j$$

Kim et al. (2003) addressed the order picking sequence problem in an automated warehouse where sections of the travelling route are fixed. The length of the warehouse is significantly longer than the width and the vehicle can only hold one part at a time. Kim et al. (2003) compared the problem with a special type of TSP.

Molnar and Lipovszki (2005) also classified the automated warehouse routing problem to be a special case of the TSP. Molnar and Lipovszki (2005) used a genetic algorithm with Pareto elitist based selection to solve the multi-criteria optimisation problem. Molnar and Lipovszki (2005) defined the genetic algorithm parameters as follows: warehouse blocks, warehouse aisles, size of the population, number of generations, and the probability of permutation. The genetic algorithm had multiple objectives, including:

- Determining the optimal number of vehicles;
- Determining the picking sequence with minimum cost;
- Minimising labour costs;
- Minimising earliness and tardiness; and
- Minimising resource utilisation.

Ratliff and Rosenthal (1983) developed an algorithm that was fast enough to be used in a realistic warehouse and it could be run on a general computer. The experimental algorithm code was developed and tested on a 50-aisle warehouse without collision avoidance. Ratliff and Rosenthal (1983) noted that the number of items in the warehouse had little effect on the solution time.

The methodology used by Ratliff and Rosenthal (1983) to solve the order picking problem and minimise the travelling time and distance can be explained as follows: Consider a warehouse with m number of parts to be picked. The warehouse configuration can be related to a graph $\boldsymbol{G}$. The graph $\boldsymbol{G}$ consists of vertices v_{0a} for *Gate 1* and v_{0b} for *Gate 2* respectively, a vertex v_i between each product $i \in \{0, 1, 2, 3, ..., m\}$, and vertices a_j and b_j at the end of each aisle. In the graph $\boldsymbol{G}$, there are connections between any two vertices and an adjacent location in the warehouse by an unlimited number of arcs, with the length of the direct distance between the two locations. For an order picking tour to be feasible in graph $\boldsymbol{G}$, it must include at least one $v_i \in \{0, 1, 2, 3, ..., m\}$.

Ratliff and Rosenthal (1983) proved the theory behind the minimum length tour using sub-graphs of $\boldsymbol{G}$. Ratliff and Rosenthal (1983) discussed the tour sub-graph as a specialisation of the theorem on Eulerian graphs (Christofides, 1975). Details regarding the theory and proofs can be seen in Ratliff and Rosenthal (1983).

3.3 Automated routing robots and automated guided vehicles

An automated warehousing system consists of a variety of computer-controlled systems and processes that can make decisions without human interference. Such systems are used to eliminate human error and for expediting processes. Some commonly found elements used in an automated warehouse include automated guided vehicles (AGVs), a mobile vehicle that can follow a set route using radio waves, cameras, magnets and lasers for navigation.

Research about automated guided vehicles (AGVs) focuses more on addressing network optimisation problems and less on developing methodological approaches to make it more sustainable in an economic and social environment. The application of AGVs can be found throughout all branches of industry, including automotive, printing, pharmaceutical, metal, food, aerospace industries and port facilities. The interest in AGVs is reflected in the sales figures of automated guided vehicle system (AGVS) sales which peaked in 2006 (Schulze et al., 2008).

The purpose of implementing AGVs is to increase the efficiency of warehouse operations and reduce warehouse costs. In order to achieve an efficient flow of AGVs, path planning is crucial as path planning will determine the time it takes for AGVs to move from one point to another. Path planning also aids in avoiding unnecessary congestion and collisions.

Schulze et al. (2008) mentioned that automation of transportation in the production, trades and service industries is critical when optimising logistic processes. Automated guided vehicles provide several benefits when implemented, costs and reliability being two of the main advantages. Vivaldini et al. (2009) studied the use of vehicle automation using forklifts and found that the use of automation improves both cost and time. AGVs also provide unmatched flexibility when integrated into an existing or changing environment. Vivaldini et al. (2009) also highlighted the significant success of AGVs regarding the repeatability of tasks.

Schulze et al. (2008) introduced the main components of an AGVS. A visual representation of an AGVS can be seen in Figure 8:

- **Vehicles**. Vehicles form the heart of an AGVS. Vehicles perform the actual activities in the system. Vehicles have to be designed specifically for the purpose of the system. Vehicles can differ in payloads, equipment, navigation systems, drive configuration, size and other aspects;

- **Stationary control system**. The task of the stationary control system is to administrate the transportation orders, optimise the schedules and communicate with other control systems. The stationary control system also provides auxiliary functions such as graphical visualisations and statistical analyses to the customer;
- **Peripheral system components**. The peripheral system components represent the counterparts to various items of on-board equipment of vehicles e.g. battery loading stations; and
- **On-site system components**. Aspects of the site's structural design that affect the AGVs, e.g. gates.

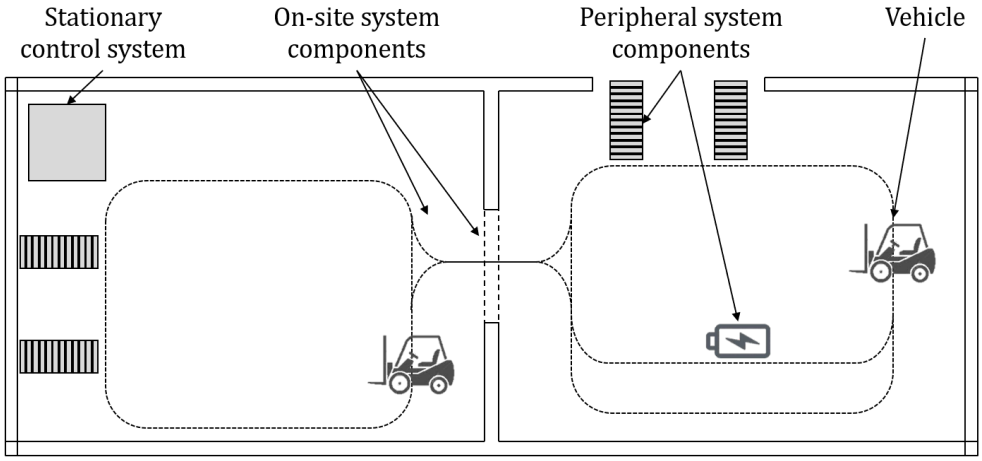

Figure 8: Automated guided vehicle system

Vivaldini et al. (2016) presented the methodology of estimating the minimum number of AGVs required to satisfy a number of tasks in a given time window. In order to achieve the minimum number of AGVs, typically the objective is to minimise the travel times of AGVs. Different algorithms were tested including the shortest job first algorithm and the tabu search metaheuristic. In order to obtain a collision-free system an enhanced Dijkstra algorithm was used for conflict-free routes. The methodology was tested on two examples of industry warehouses and the results can be seen in more depth in Vivaldini et al. (2016).

AGVs have matured to such an extent that they can add great value to a supply chain with the focus on sustainability (Bechtsis et al., 2017). Some AGVs are capable of planning their own movements to execute a picking or binning process. The planning of their movements takes into account the sharing of aisle space with other manual or automated vehicles (Jacobus et al., 2015). AGVs keep to planned speed limits; they are able to stabilise their loads using stability control and they can determine when to stop or move. AGVs are able to make human-like decisions at intersections to merge into the traffic flow. AGVs use sensors to identify static and dynamic obstacles, and given the feedback from the sensors, the AGV can determine whether to stop and wait until the obstacle is removed or avoid the detected obstacle (Jacobus et al., 2015).

De Ryck et al. (2020) provided an overview of the most recent AGV control algorithms and techniques. They divide AGV control tasks into five main areas; namely, task allocation, localisation, path planning, motion planning, and vehicle management. This dissertation focuses mainly on task allocation and path planning.

Task allocation (Khamis et al., 2015) is concerned with determining which robot performs which tasks, e.g. the picking and binning of parts at certain locations. Path planning is concerned with generating the shortest obstacle-free path from the point of origin to destination and back (Jabbarpour et al., 2017; Lee et al., 2018; López-González et al., 2020).

Another important distinction to make is the difference between centralised and decentralised AGV architectures. Centralised systems make use of global information to find an optimal or near optimal solution for the AGV system. This optimisation is typically performed before initiation of the first tasks and is static in nature. An example is allocating tasks and planning the best paths for all robots in the system. A decentralised system makes use of local information and the individual robots have built-in intelligence to make their own decisions as the system changes dynamically (De Ryck et al., 2020; Draganjac et al., 2020; Chen et al., 2018). This dissertation focuses on the centralised case where the system needs to be re-optimised each time a significant change occurs.

An investigation into existing research related to centralised AGV architectures showed that the requirement for **collision avoidance** and **multiple objectives** adds significant complexity to the problem. In a warehouse the aisle and shelf layout need to be considered and collisions between multiple robots need to be avoided.

Collision-avoidance systems and strategies are used to predict the risks of a collision happening and a strategy to avoid the possible collision. These strategies include the guidance for vehicles to wait, decelerate, or avoid a path. Collision-avoidance systems are able to perform actions autonomously by applying a set of predefined rules and restrictions.

3.4 Single objective robot path planning and task allocation literature

This section discusses task allocation and path planning strategies and approaches found in literature. Han et al. (1987) defined two main approaches for sequencing picking and binning processes, namely:

1. Define the high turnover products, sequence the products that are more frequently required first, and only after the frequent products have been sequenced, select the rest of the products.
2. Allow for re-sequencing. Every time a new product is required to be sequenced, apply heuristics according to due dates or priorities.

The two approaches have different goals when solving the sequencing problem. Defining the high turnover products will allow the model to assign a priority to these items. The benefit of using priorities can be to ensure that the more frequently bought products always meet their requirements. The downside of priorities is that it sometimes neglects the products that do not have as high priorities. The second approach of re-sequencing allows the algorithm to re-sequence products if a better solution is found, but on the other hand it also creates an exponentially larger solution space to be solved. Solving large solution spaces tends to be computationally expensive and it is practically infeasible, thus limiting the number of times re-sequencing will ensure the solution space does not grow too large to solve.

Fanti et al. (2015) considered a zone controlling guide path network for assigning tasks to multiple AGVs. The controlling network guided the AGVs on respective paths to avoid collisions and deadlocks. All vehicles in the system applied a discrete consensus algorithm to locally minimise the global cost of reaching their destination. The consensus protocol was solved using local integer linear programming.

The AGVs moved along the network to reach their respective destinations using a decentralised coordination algorithm to avoid collisions and deadlocks. Fanti et al. (2015) showed that the coordination procedure can guarantee the avoidance of collisions and deadlocks.

A simultaneous task allocation and motion coordination (STAMC) approach was presented by Liu and Kulatunga (2007), Liu and Kulatunga (2007) who applied two metaheuristics; namely, simulated annealing and an auction algorithm. The proposed approach was able to solve the scheduling, planning, and collision-avoidance problems simultaneously. Additional benefits of the STAMC approach is the improvement to use bottleneck areas, the ability to handle dynamic traffic-rules, and to avoid a deadlock. The effectiveness and efficiency of the STAMC approach was demonstrated using simulation results.

Draganjac et al. (2020) presented a method that can be used to coordinate free-ranging AGVs in real-world scenarios. The primary aim of the study was to enhance current multi-vehicle transportation systems. The presented method relies on a centralised controller which has a predetermined network of available paths. The improvements Draganjac et al. (2020) addressed include; better scalability, a more flexible system and the introduction of a decentralised controlling algorithm. The algorithm proposed by Draganjac et al. (2020) integrates path planning and navigation. Each vehicle was responsible for planning its own shortest feasible route towards the destination location. The vehicle followed the shortest route until there was a conflict with another vehicle; the first vehicle then resolved the conflict situation with the other vehicle and continued on the shortest path. The navigation strategy relied on a private zone mechanism, ensuring reliable collision avoidance. The scalability was improved by allowing negotiations with other vehicles in a limited communication radius. The experimental results confirmed that the method was highly scalable and the coordination method was highly flexible with changes in the environment.

Vivaldini et al. (2010) presented research on robotic forklifts for an intelligent warehouse system. The focus was on the three key routines that determined the behaviour of the AGV; namely, the algorithm used for the overall routing of the AGV through the typographically mapped warehouse for the objective of minimum time or distance, the local path-planning algorithm for the node-specific route planning and optimisation, and lastly an auto localisation algorithm to estimate the AGV's actual position. The algorithm was able to resolve the traffic congestion and avoid collisions before the AGVs started travelling.

Kim et al. (2003) introduced a clustering-based sequencing algorithm to solve the automated warehouse routing problem. The clustering-based sequence algorithm has three steps: clustering locations based on their horizontal positions, solving each cluster individually, and combining the sub-tours. The three steps of the clustering algorithm developed were:

Step 0: Sort locations in increasing order of x-coordinates

Step 1: Set current index cluster = 1. Select location with smallest x-coordinate and place into current cluster.

Step 2: From the remaining unselected locations check that the horizontal movement time between that location and any other from the current cluster, is less than the vertical movement times of the locations. If so, place the location into the current cluster. Repeat **Step 2** until there are no more such locations.

Step 3: If there are no more locations remaining to select, **STOP**.
Otherwise set the current index cluster = current index cluster +1 and go to **Step 1**.

Vivaldini et al. (2009) proposed an algorithm that produced the optimal routes for AGVs. The algorithm was able to solve real-world scenarios with conflict-free paths amidst the presence of obstacles. The algorithm proposed by Vivaldini et al. (2009) was based on Dijkstra's shortest path method. The algorithm allowed AGVs to execute tasks by starting at an initial position and travelling to a pre-established position using a minimum path methodology. The path, however, formed a continuous sequence of positions that the AGV travelled along. The continuity of the path allowed for real-world decision-making to overcome obstacles and congestion. Vivaldini et al. (2009) validated the efficiency of the proposed algorithm using computer simulations on different real-world scenarios.

Traditional routing algorithms typically use static methods for calculating the most optimal routes. In a real-world scenario static calculations can only be adjusted slightly to accommodate unplanned occurrences (Klaas et al., 2011). Klaas et al. (2011) aimed to develop a routing algorithm that would be more reactive to real-world scenarios. It needed to plan routes in smaller steps, creating the routing path incrementally rather than holistically. The incremental steps made the algorithm more likely to find a feasible solution in real-world scenarios like collision avoidance and deadlocks. Machine learning was also used in a discrete simulation process to learn from experiments. The algorithm was tested and the results yielded an algorithm that was naturally robust against delays and failures.

Fuzzy logic control combined with a potential field were used for AGVs' path planning by Wu et al. (1999). The potential field was created using the top view of an environment. The chamfer distance transform was used to build the imaginary potential field required. The potential field method was used to calculate the force between the vehicle and the closest obstacle (Wu et al., 1999). The resultant forces then guided the AGV to the destination position. In a deadlock situation (where no vehicle is able to move in the desired direction), a fuzzy logic controller was proposed to adjust the direction of the AGV. Wu et al. (1999) performed a series of simulations and the results show that the fuzzy logic controller was able to successfully navigate the AGV out of deadlock situations.

Chen et al. (2018) used the artificial potential function (APF) method to provide a solution to the path planning problem for multi-robot systems. The improved APF method was able to solve the following problems: obstacle avoidance, falling into a local minimum, not reaching the target fitness value, collision avoidance and avoiding traffic congestion. The following methods were used to solve the listed problems; the wall-following method was used to eliminate the risk of falling into a local minimum, the APF method was used to ensure that the target fitness value was reached, and the priority strategy along with a collision-avoidance algorithm were used to solve congestions and collisions. Multiple simulations were performed and the results confirmed the validity and feasibility of the methods used.

Zhang et al. (2018b) proposed a collision-free routing method for AGVs based on collision classification. The warehouse was divided into areas. After the areas were defined the algorithm performed route planning for each AGV. The initial route of each task was determined using the improved Dijkstra's algorithm. The initial routes were shared and interpreted by the centralised controller. The centralised controller detected potential collisions by comparing the locations of the AGVs at each time interval. The collisions were then classified using the following criteria:

- Head-on collision: a head-on collision happened when two AGVs travelled on the same path, at the same time but in opposite directions.
- Cross collision: a cross collision happened when two AGVs wanted to enter the same intersection at the same time.

- Node-occupancy collision: the node-occupancy collision happened when an AGV was occupying a node and another AGV wanted to pass through that node or use the same node.
- Shelf-occupancy collision: the shelf-occupancy collision could happen when multiple AGVs attempted to service the same picking location but from different directions.

After the collisions had been identified and classified, the centralised controller provided a collision-free solution.

Pamosoaji and Hong (2011) presented an algorithm that was able to generate collision-free routes in a multi-vehicle system. Three-degree Bezier curves were used to form the basic pattern of the routes. The presence of known static obstacles and collisions with another vehicle were considered in the solution. The PSO algorithm was used to optimise the routes, as the minimum time required.

Juntao et al. (2016) proposed a path planning method with collision avoidance for AGVs. The improved ant colony algorithm combined with the grid method with time windows was used to determine the optimal path for the AGVs. Simulation results indicated that the proposed method and algorithm can identify the optimal solution for the path planning problem. The solution was able to plan the routes of multiple AGVs without any collisions.

While solving the path planning problem, Yuan et al. (2016) also focused on the reduction of traffic congestion as well. A collision-free path planning method based on the A* improved algorithm was presented. The method to avoid collisions and congestion was to add a penalty value when multiple AGVs shared the same path. The penalties were set according to the different types of traffic congestion and collisions. The collision-free path was then constructed by combining the improved A* algorithm and collision resolution heuristics. Simulation results showed that the method presented can improve the warehouse's activity efficiency and bring relief to the traffic congestion in the system.

Zhang et al. (2018a) developed a time window method to find the optimal paths for a multiple AGV system. The A* algorithm was used to find the optimal paths for the multi-AGV system. The algorithm developed was tested in a simulation experiment and the results showed that the algorithm can effectively solve conflicts between AGVs and that the algorithm was reliable in a real-world environment.

Zhang et al. (2017) proposed a conflict-free route planning approach for AGVs in a warehouse system. Dijkstra's algorithm was used to determine the candidate routes for the proposed algorithm. Zhang et al. (2017) used the method of conflict classification to resolve the potential conflicts. These classifications include the cross-conflict when two AGVs want to enter the same intersection, the path-occupancy conflict where one AGV is occupying the path which another AGV wants to occupy, and the head-on conflict where two AGVs meet on the same path but are heading in opposite directions. After the conflict was classified, a self-adaptive strategy was used to resolve the conflict. The efficiency of the proposed conflict-free route planning approach was verified using simulation techniques.

Jabbarpour et al. (2017) proposed an ant-based path planning approach. The objective was to find the shortest collision-free path. The method was validated using simulation tools and the performance was compared to the ant colony optimisation, genetic algorithm, and the particle swarm optimisation algorithm. The results showed that the ant-based path planning approach outperformed the existing methods.

Jacobus et al. (2015) developed an algorithm where each AGV received their current or starting position and their final destination. The path for each AGV was calculated incrementally using three steps (see Figure 9):

1. The Euclidean shortest path from starting position to destination was calculated for each AGV. Each path was cut off at the point where a collision would occur. The AGV moved along this shortest path at a constant speed until the cut-off point was reached, minus an offset.

2. Rerouting was used to avoid the collision according to predefined rules. The rules were determined for the current system state by accessing the knowledge base. The knowledge base consists of situations previously encountered during the offline simulation. The scenarios encountered in the offline simulation have best option rules given to each of them, thus the rule used in the algorithm's scenario received the optimal re-routing rule from the knowledge base.

3. If the rerouting was successful, return to Step 1.

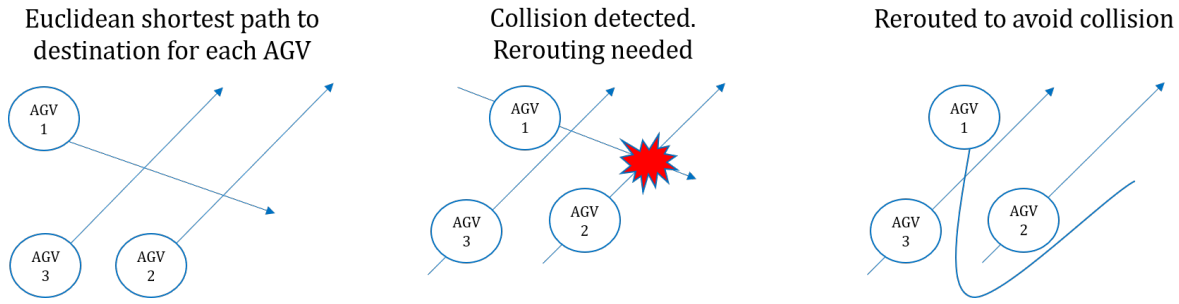

Figure 9: Incremental path planning for AGV

Because of the incremental calculation, the algorithm was dynamic. Additional stops on route, modifications to routes, and unforeseen delays could be accommodated.

Vis et al. (2001) developed a minimum flow algorithm to determine the number of AGVs required at a semi-automated container terminal. Vis et al. (2001) categorised the two main activities of the AGVs as loading and unloading of containers. The algorithm developed by Vis et al. (2001) was a strongly polynomial time algorithm which was able to solve a scenario where the containers were available for transport at known time instances. The model and algorithm were developed to determine the number of AGVs needed to move all the containers within a given time frame. The model did not allocate containers to specific vehicles, which was done by operational personnel. The problem was relaxed to an extent where the availability of containers was at predetermined time instances.

Menon et al. (1988) presented the problem of mobile robot navigation and planning in an automated warehouse. A generalised layout was considered and the objective of the path planning algorithm was to find the shortest distance while avoiding any collisions. The developed algorithm's process flow was closely related to the part sequencing and allocation problem.

The process flow was as follows:

1. Create a global map of the warehouse and the nodes that can be used for movement.
2. Assign the warehouse tasks to the number of robots available in the system.
3. Introduce static constraints (if they exist).
4. Path planning from current node to destination node.
5. Create node arrival time charts for each robot. Alter the arrival times to avoid collisions.
6. Initiate movement and scan the next node's status.
7. Is there a risk of collision or obstacle? If yes, update constraints and return to step 4.
8. Move towards destination node and after each movement re-evaluate from step 6 onwards.
9. Stop when destination node is reached.

Kovács and Kóczy (1999) investigated the use of fuzzy logic algorithms as guidance systems, path tracking, and collision-avoidance strategies. The main purpose of the guidance control system was to keep record of the paths used. Secondly, the strategy had to avoid any type of collision without risking the chance of losing the guided path. The strategies were developed using the actual operator's control actions. The control actions translated to heuristics which formed the fuzzy rule database. Kovács and Kóczy (1999) concluded that the approximate fuzzy reasoning method was an efficient approach for designing a direct fuzzy logic control system.

Lee and Wang (1994) also applied the fuzzy logic approach as a collision-avoidance strategy for AGVs. Static obstacles as well as dynamic obstacles with unknown velocities were considered. Human-defined strategies for collision avoidance were modelled as fuzzy logic rules. Fuzzy logic was used to guide an AGV from the AGV's initial position to the destination position without colliding with other vehicles or obstacles. The method proposed by Lee and Wang (1994) can also be used for the navigation of multiple AGVs. A simulation model was used to show the feasibility of the proposed fuzzy logic approach.

Collision and deadlock problems that may occur when routing an AGV in real time were investigated by Möhring et al. (2005). Möhring et al. (2005) developed and presented an algorithm that was able to avoid collisions and deadlocks at the time of route computation. The algorithm was able to produce conflict-free solutions. The algorithm's real-time calculations were able to determine the shortest route with time windows, followed by a readjustment of the time windows. Both were done in polynomial time. To decrease the computational time, a goal-orientated search technique was used (Möhring et al., 2005). The algorithm's performance was compared to a static routing approach. It was notable that the algorithm proposed was easy to implement and it was not computationally complex.

Watanabe et al. (2001) proposed two methods to provide intelligence to AGVs. For an AGV to drive autonomously, two problems need to be solved; namely, the navigation of multiple AGVs and collision avoidance. Watanabe et al. (2001) considered a new method based on the feature scene recognition and acquisition for solving the navigation problem. The navigation route of an AGV was learnt by the use of Q-learning (a reinforcement learning algorithm). Addressing the second problem, collision avoidance, Watanabe et al. (2001) described the problem as a mutual understanding of behaviours between multiple AGVs. Combining these two methods allowed for the routing of multiple AGVs autonomously in a factory.

Digani et al. (2014) attempted to solve the obstacle avoidance problem by proposing an automated algorithm that was capable of building new obstacle-free trajectories that deviate from the original path. The newly built path had to comply with the dynamic and kinematic constraints of the AGV. The new path was generated using polar spline curves, lane change curves and line segments. The algorithm proposed by Digani et al. (2014) had the following objectives:

- Computing a path that guaranteed obstacle avoidance;
- Defining a path that was admissible; and
- Guaranteeing that once an obstacle had been passed, the AGV returned to the original path.

The detailed algorithm can be seen in Digani et al. (2014). Given the objectives above, the algorithm had a defined sequence of steps to follow:

1. Safety check.
2. Leave the road map.
3. Avoid the obstacle.
4. Return to the original path.

The proposed method was validated by testing various scenarios in a simulation model.

Ganapathy et al. (2010) developed an improved ant colony optimisation algorithm that could navigate a robot in a warehouse while avoiding any collisions. Systematic testing and simulations showed that the algorithm was capable of finding feasible solutions to the real-world problem. Uriol and Moran (2017) also used the ant colony optimisation algorithm to solve the path planning problem of robots in complex environments. The results also showed that the ant colony optimisation algorithm was able to find optimal or near-optimal shortest paths with collision avoidance. A metaheuristic-based ant colony algorithm was also used by Kulatunga et al. (2006). The results showed that the ant colony optimisation algorithm achieved slightly better performance than the results of the simulated annealing algorithm.

A summary of collision-avoidance articles with their techniques, objective functions and constraints is provided in Table 3.4.

Table 1: Research that includes collision avoidance

Author	Algorithm	Objective	Constraints
Draganjac et al. (2020)	Motion coordination strategy algorithm	Minimum cost function	Obstacle avoidance
Haiming et al. (2019)	A* shortest-path method with time windows	Minimum cost function	Collision avoidance
Zhang et al. (2018b)	Improved Dijkstra's algorithm	Feasible solutions	Collision avoidance
Chen et al. (2018)	Shortest path theory and algorithm	Shortest distance	Obstacle avoidance
Zhuang et al. (2018)	Jump point search (JPS)	Shortest distance	Collision avoidance
Dhawale et al. (2018)	Gaussian mixture model (GMM)	Minimum time	Collision avoidance
Liu et al. (2017)	Novel swarm robot simulation platform	Minimum time and distance	Collision avoidance
Gochev et al. (2017)	A* shortest-path method	Minimum time	Collision avoidance
Pamosoaji and Hong (2011)	Particle swarm optimisation (PSO)	Minimum time	Collision avoidance
Liu and Kulatunga (2007)	Simultaneous task allocation and motion coordination.	Minimum time	Scheduling vehicles.
Tan (2006)	Rapidly exploring random tree (RRT) algorithm	Feasible solutions	Collision avoidance
López-González et al. (2020)	Genetic algorithm	Feasible solutions	Collision avoidance
Cesarone and Eman (1989)	Dynamic programming	Shortest distance	Collision avoidance
Juntao et al. (2016)	Ant colony algorithm	Minimum time	Collision avoidance
Jan et al. (2008)	Higher geometry maze routing algorithm	Shortest distance	Collision avoidance
Yuan et al. (2016)	A* shortest-path method	Shortest distance	Collision avoidance
Vivaldini et al. (2016)	Dijkstra's shortest path method	Minimum time	Collision avoidance
Hennes et al. (2012)	Adaptive Monte Carlo	Minimum time and distance	Obstacle avoidance
Yan et al. (2017)	Digraph model	Feasible solution	Collision avoidance
Menon et al. (1988)	Path planning algorithm	Shortest distance	Collision avoidance
Zhang et al. (2018a)	A* shortest-path method	Minimum time	Collision avoidance
Zhang et al. (2017)	Dijkstra's shortest path method	Shortest distance	Collision avoidance
Truong et al. (2018)	Continuous cluster method	Minimum time and distance	Collision avoidance

Table 1: Research that includes collision avoidance

Author	Algorithm	Objective	Constraints
Dong et al. (2016)	Cultural-genetic algorithm	Shortest distance	Collision avoidance
Digani et al. (2019)	Quadratic program	Minimum time	Collision avoidance
Li et al. (2018)	Dynamic programming	Shortest distance	Collision avoidance
Ó Duinn (1994)	Lee's algorithm	Minimum time and distance	Collision avoidance
Jabbarpour et al. (2017)	Green ant algorithm	Shortest distance	Collision avoidance
Chen and Liu (2019)	Ant colony algorithm	Shortest distance	Obstacle avoidance
Lee et al. (2018)	Genetic algorithm with a direction guided factor	Shortest distance	Collision avoidance
Ganapathy et al. (2010)	Improved ant colony optimisation	Shortest distance	Obstacle avoidance
Uriol and Moran (2017)	Ant colony optimisation algorithm	Shortest distance	Obstacle avoidance
Larsen et al. (2017)	Sampling-based and computational intelligence methods	Shortest distance	Collision avoidance
Fazlollahtabar and Hassanli (2018)	Network simplex algorithm	Minimum time	Collision avoidance
Wang and Banitaan (2018)	A partitioning-based approach	Low cost paths	Collision avoidance
Suchi and Vincze (2014)	Metaheuristic search strategies	Minimum time	Collision avoidance
Kulatunga et al. (2006)	Ant colony optimisation	Minimum time	Collision avoidance
Dao et al. (2016)	A multi-objective model	Shortest distance	Collision avoidance
Pal and Sharma (2013)	Swarm intelligence	Shortest distance	Collision avoidance
Sahoo et al. (2011)	Honey bee mating optimisation algorithm	Shortest distance	Collision avoidance

The article on order picking in a rectangular warehouse by Ratliff and Rosenthal (1983) defined the fundamental item retrieval problem as the order picking problem. Ratliff and Rosenthal (1983) described the same type of picking problem addressed in this dissertation: the collection of parts from a dispatch area and returning to the dispatch area. The objective function is also very similar. Ratliff and Rosenthal (1983) set the objective to minimise the time travelled or distance travelled. The assumption of a vehicle only collecting and binning one item at a time is also mutual. Ratliff and Rosenthal (1983) stated that for an arbitrary configuration of a warehouse, the order picking problem can easily be recognised as a variant of the TSP, although it can be very difficult to solve.

The optimisation of both item locations and routing for an order picking vehicle has received significant attention in literature. Beroule et al. (2017) investigated the order picking problem in a hospital pharmacy warehouse. The purpose of the article was to simultaneously solve the routing improvement problem and the location allocation problem. Beroule et al. (2017) provided the evidence that the problem can be solved using an exact mathematical model. The solution was obtained by the use of mixed integer programming. The mixed integer model could be extended to a linear version. The disadvantage was the running time required to find the optimal solution. Although it was possible to solve the problem exactly, the following assumptions were made to reduce the problem to an exact solvable case:

- Collisions between robots are not considered in this problem;
- The order robots are never overloaded;
- The picking operations start and end at the same geographical point; and
- Only one class of an item may be placed at any one location.

Because of the running time required by an exact method, a genetic algorithm (GA) and dedicated heuristic were presented to form a hybrid GA for solving the problem. Other solution methods worth mentioning from the research by Beroule et al. (2017) include the use of a hybrid of a dedicated heuristic and the TSP, the hybrid GA, and the sole use of a GA. In order to understand the working of a GA in this context an example is shown as presented in Beroule et al. (2017):

- $n = number\ of\ items\ (and\ locations)$;
- $m = number\ of\ picking\ lists$;
- $\boldsymbol{C}_t = set\ of\ items\ contained\ in\ a\ list\ t$;
- $D_{k,l} = distance\ between\ a\ location\ k\ and\ l$;
- $x_{i,k} = \begin{cases} 1\ if\ item\ i\ is\ at\ location\ k \\ 0 otherwise \end{cases}$
- $l_{i,j,t} = \begin{cases} 1\ if\ list\ t\ implies\ to\ go\ from\ item\ i\ to\ j\ when\ picking \\ 0 otherwise \end{cases}$
- $u_{i,t} = picking\ order\ of\ item\ i\ in\ list\ t$.

The aim of the model developed by Beroule et al. (2017) was to minimise the distance travelled. Thus the objective function was formulated as:

$$Minimise\ Z = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{t=1}^{m} x_{i,k} \times x_{j,l} \times l_{i,j,t} \times D_{k,l} \tag{39}$$

The model was subject to:

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad 1 \leq i \leq n \tag{40}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \quad 1 \leq j \leq n \tag{41}$$

$$\sum_{j \in C_t} l_{i,j,t} = 1 \quad i \in C_t\ 1 \leq t \leq m \tag{42}$$

$$\sum_{i \in C_t} l_{i,j,t} = 1 \quad j \in C_t\ 1 \leq t \leq m \tag{43}$$

$$l_{i,j,t} = 1 \quad 1 \leq t \leq m \tag{44}$$

$$u_{1,t} = 1 \quad 1 \leq t \leq m \tag{45}$$

$$u_{i,t} - u_{j,t} + nl_{i,j,t} \leq n - 1 \quad 1 \leq i \leq n, \quad 2 \leq j \leq n, \quad 1 \leq t \leq m, \quad i \neq j \tag{46}$$

$$x_{1,1} = 1 \tag{47}$$

Equations (40) and (41) force each product to be allocated to a location. In the study of Beroule et al. (2017), the location number 1 is considered as the start and end of the picking and binning operation. For this reason, Equation (47) forces the depot to be at location 1.

Equations (42) and (43) ensure that each item is preceded and followed by another item. Note that each picking list starts and ends with the depot (location 1). Equation (44) is used to prevent duplicating a picking order. Lastly, Equations (45) and (46) prevent the picking lists from containing sub-tours.

The equations discussed above can be solved with an exact solver, but the complexity of the objective function drastically increases the computational time. Beroule et al. (2017) found that the computation time may reach up to an hour to solve a problem with only ten products.

In order to implement the GA, the solution representation had to be calculated. Beroule et al. (2017) presented each solution representation as shown in Equation (48):

$$\boldsymbol{P} = \begin{cases} \boldsymbol{L} = \{L_1, L_2, ..., L_m\} \\ \boldsymbol{X} = \{X_1, X_2, ..., X_n\} \end{cases} \tag{48}$$

Each feasible solution $\boldsymbol{S}$ was composed of a set of order picking lists ($\boldsymbol{L}$) and locations ($\boldsymbol{X}$). A picking list consisted of orders to be picked (i) giving the final list $\boldsymbol{L}_i$ of orders to be picked in the respective list. The location vector $\boldsymbol{X}$ matched the location for item i, giving each item a location number X_i. The GA in Beroule et al. (2017) started by creating the initial population composed of randomly created solutions. The initial population evolved by means of crossover and mutation. The crossover allowed for parent solutions to be merged into a child solution. During each iteration a portion of the current population was selected (based on the crossover rate) for crossover. It was performed by two crossing operators; the first operator identified the robust parts of each list $\boldsymbol{L}_i$ and $\boldsymbol{L}_i^{'}$ from the two parent solutions.

The non-robust items were selected randomly and the location vector $\boldsymbol{X}$ was randomly selected from either one of the two parents. If two solutions had to be paired, a crossover operator was randomly chosen. Examples of each operator are shown in Equations (49) and (50):

$$\begin{cases} \boldsymbol{L}_i = \{2, \underline{\mathbf{5}}, 8, 6, \underline{\mathbf{7}}, \underline{\mathbf{4}}\} \\ \boldsymbol{L}_i^{'} = \{8, \underline{\mathbf{5}}, 6, 2, \underline{\mathbf{7}}, \underline{\mathbf{4}}\} \end{cases} \rightarrow \{6, \underline{\mathbf{5}}, 2, 8, \underline{\mathbf{7}}, \underline{\mathbf{4}}\} \tag{49}$$

$$\begin{cases} \boldsymbol{X} = \{\underline{\mathbf{1}}, 4, 6, \underline{\mathbf{2}}, 3, 7, \underline{\mathbf{5}}, 8\} \\ \boldsymbol{X}^{'} = \{\underline{\mathbf{1}}, 6, 7, \underline{\mathbf{2}}, 8, 4, \underline{\mathbf{5}}, 3\} \end{cases} \rightarrow \{\underline{\mathbf{1}}, 7, 3, \underline{\mathbf{2}}, 4, 6, \underline{\mathbf{5}}, 8\} \tag{50}$$

The mutation process in the GA was a mechanism that allowed the solution to be modified by itself. During the iterations of the GA a part of the current population was selected to be mutated. The part of the population to be mutated was defined by the mutation rate given as input parameter. As with the crossover process, the mutation process also consisted of two operators. The first operator randomly selected two items in the list set $\boldsymbol{L}$. The first operator then changed the items selected. The second operator randomly selected two elements of the location vector $\boldsymbol{X}$ and changed them. When the solution had to be mutated a mutation operator was randomly chosen. Examples of the two operators are shown in Equations (51) and (52):

$$\boldsymbol{L}_i = \{2, \underline{\mathbf{5}}, 8, \underline{\mathbf{6}}, 7, 4\} \rightarrow \{2, \underline{\mathbf{6}}, 8, \underline{\mathbf{5}}, 7, 4] \tag{51}$$

$$\boldsymbol{X} = \{1, \underline{\mathbf{4}}, 6, 2, \underline{\mathbf{3}}, 7, 5, 8\} \rightarrow \{1, \underline{\mathbf{3}}, 6, 2, \underline{\mathbf{4}}, 7, 5, 8\} \tag{52}$$

The 'new' solutions were evaluated and the best ones were selected to create a new population of the same size. The reproduction process continued until the desired outcome had been reached or the iteration limit had been reached. The best solution of the final generation's population was considered to be the output of the GA.

Beroule et al. (2017) improved on the normal GA by creating a hybrid GA. Beroule et al. (2017) introduced the TSP and a dedicated heuristic (DH) to create the initial population. Beroule et al. (2017) compared the hybrid GA in terms of solution fitness and computational time. The size of the warehouse consisted of 400 item locations; the picking lists were numbered 1 to 25, the number of items in a list were 10 to 90 with increments of 10. The GA was given the following parameters:

- Initial population = 200;
- Number of iterations = 1000;
- Crossover rate = 50%; and
- Mutation rate = 30%.

The improvement of the hybrid GA (HGA) compared to the simple GA (Δ HGA/GA) and compared to the dedicated heuristic (Δ HGA/DH) were calculated using:

$$\Delta HGA/GA = \frac{FitHGA - FitGA}{FitGA} \times 100 \tag{53}$$

$$\Delta HGA/GA = \frac{FitHGA - FitDH}{FitDH} \times 100 \tag{54}$$

Beroule et al. (2017) observed the following results: the average improvement of the hybrid GA was 23% and the average improvement in the dedicated heuristic was 30%. Beroule et al. (2017) concluded by stating that the hybrid GA was very efficient compared to the classic GA even if no comparison could be made with exact solutions.

3.5 Multi-objective robot path planning and task allocation literature

A number of multi-objective algorithms have already been used to solve a version of the part sequencing and allocation problem. An investigation into multi-objective multi-robot path planning and task allocation literature where both static and dynamic collisions need to be avoided resulted in the following studies being identified as relevant:

- Chen et al. (2020) investigated path planning and task allocation for a parking robot. An improved genetic algorithm and a time-enhanced A* path planning algorithm were developed for minimising the parking lot operating cost and time cost functions. A weighted sum of the objectives was minimised.
- Majumder et al. (2021) made use of an A* path planning algorithm combined with a multi-objective teaching-learning-based optimisation algorithm for task allocation. Total completion time and total fuel consumption of a multi-robot plant inspection system were minimised.
- Majumder and Ghosh (2020) used an A* path planning algorithm combined with a multi-objective particle swarm optimisation algorithm for task allocation. Total completion time and total fuel consumption were minimised.
- Majumder and Ghosh (2020) proposed a tailor-made multi-objective GA with alternative set-up and search operators, and a reinforcement learning approach to minimise the cost and quality of robot deployment. Collisions are avoided by ensuring that different robots are sent to different areas of the work space.
- Okumuş et al. (2020) developed a D* lite algorithm for minimising the path length, elapsed time, and energy expenditure of multiple AGVs employed in a fabric-manufacturing enterprise. Task allocation was performed using a nearest neighbour algorithm.
- Sullivan et al. (2018) used a NSGA-II to minimise the total time, maximum time and maximum net gap of a multi-robot system. Network connectivity and energy efficiency were the most important considerations.
- Biswas et al. (2021) solved a multi-objective mission route planning problem where the time and cost objectives were considered simultaneously. A vectorised particle swarm optimisation algorithm was used for path planning and a nearest neighbour algorithm for task allocation. A neural network-based prediction model was also used to forecast the mission completion time.

From the limited existing literature, it can be concluded that research into the multi-objective multi-robot path planning and task allocation problem is still in its infancy. The use of a weighted sum of objectives to address multiple objectives and local search algorithms such as the nearest neighbour algorithm for task allocation, highlights significant opportunities for improvement. Finally, the exact configuration of the problem solved in this dissertation could not be found in the literature.

It is notable that a multi-objective CMA-ES has never been tested on this specific problem. Given the research described in this section, the contribution of this research includes the incorporation of collision avoidance in the robot routing multi-objective problem while using metaheuristics as solution strategies, thus to the best of the author's knowledge, this research is the first where the CMA-ES is benchmarked against the PSO, GA and DE for the multi-objective problem, given that all collisions must be avoided.

3.6 Summary

This chapter introduced warehousing activities and warehouse layout options with regard to order picking and binning in Section 3.1. Special cases of the TSP that have similarities to the robot routing and allocation problem were discussed in Section 3.2. Section 3.3 focused on the AGVs and AGVS and how warehouse automation can be feasible. The single objective robot path planning and task allocation literature were discussed in Section 3.4. After that, the multi-objective literature on the robot path planning and task allocation were discussed in Section 3.5. The next chapter introduces the single objective mathematical model and algorithm framework used to solve the single objective part sequencing and allocation problem.

Chapter 4

Single objective mathematical model and algorithm

In order to solve the single objective part sequencing and allocation problem a mathematical model was developed. The following sections contain a comprehensive description of the problem. Section 4.1 presents the detailed mathematical model. Section 4.1 also discusses the function of each constraint and the algorithm's assumptions and limitations. Section 4.2 describes the algorithm framework and its workings.

4.1 Single objective part sequencing and allocation model

In a warehouse with N number of parts, all the parts need to be picked and binned on a daily basis. The picking and binning is done using one of K robots. A robot cannot carry more than one part at a time. A robot enters the warehouse through the entrance gate, completes the job (pick or bin), and then leaves through the exit gate. The time it takes a robot to pick or bin a part is equal to the standard moving time plus the delay time for collision avoidance, the physical picking or binning time, and the time to travel to the exit gate.

The objective function of the model is to minimise the total time it takes all the robots to complete their picking and binning processes by changing the sequence and allocation of parts to be picked and binned. The sequence has a significant influence on the total picking and binning time. Each sequence has different collision avoidance scenarios that must be incorporated. Changing the sequence affects the number of collisions to avoid and also influences the total picking and binning time.

The following symbols need to be denoted:

$$I \triangleq \text{The number of parts in the warehouse; and}$$

$$K \triangleq \text{The number of robots in the system.}$$

This chapter has been published in Croucamp and Grobler (2021).

$$x_{ijk} = \begin{cases} 1 \text{ if robot } k \text{ is binning or picking part } i \text{ immediately before part} \\ \quad j \text{ where } i, j \in \{1, ..., I\} \text{ are the parts to be picked and} \\ \quad i, j \in \{I+1, ..., 2I\} \text{ are the parts to be binned,} \\ 0 \text{ otherwise} \end{cases}$$

$w_{ijk} \triangleq$ Time added to avoid collision if part i is picked or binned immediately before part j by robot k

$t_i \triangleq$ The transport time without delays of part i from entrance to exit of warehouse

$n_i \triangleq$ Starting time of new task for part i from the entrance gate

$p \triangleq$ Process time for picking or binning incurred for each part

$$\text{Minimise } Z = \max_{1 \leq i \leq 2I} \{n_i\} \tag{55}$$

The model is subject to:

$$\sum_{i=1}^{2I} \sum_{k=1}^{K} x_{ijk} = 1 \quad \forall j \in \{1, ..., 2I\} \tag{56}$$

$$\sum_{j=1}^{2I} \sum_{k=1}^{K} x_{ijk} = 1 \quad \forall i \in \{1, ..., 2I\} \tag{57}$$

$$\sum_{i=1}^{2I} x_{ipk} - \sum_{j=1}^{2I} x_{pjk} = 0 \quad \forall p \in \{1, ..., 2I\}, k \in \{1, ..., K\} \tag{58}$$

$$\sum_{k=1}^{K} \sum_{i=1, i \neq j}^{2I} x_{ijk}(n_i + p + w_{ijk} + t_i) \leq n_j \quad \forall j \in \{1, ..., 2I\} \tag{59}$$

$$\begin{aligned} t_i, n_i, p, w_{ijk} &\in \{0, R\} \\ x_{ijk} &\in \{0, 1\} \end{aligned}$$

The objective function of the model (Equation (55)) is to minimise the maximum time it takes to complete all the picking and binning jobs for all the robots. Equations (56) and (57) ensure that all parts are picked and binned respectively, ensuring that each robot can only pick and bin one part at a time, and also that no two robots pick or bin the same part at the same time.

Equation (58) enforces continuity in the model, ensuring that the next part is picked or binned. The different delay times that can be incurred, depending on the number of robots on the same route, are calculated using Equation (59). If there are robots on the route the equation will add the respective delay time given the position of the other robots. This constraint is used to avoid collisions in the warehouse.

The limitations and assumptions of the model include:

- The entrance and exit gate, respectively, can only be used for entering and exiting the warehouse;
- The aisles can only accommodate one-way traffic;
- A robot uses one time step to move from one pickup point to the next;
- The picking and binning time is assumed to be constant;
- If a robot exits the warehouse it is automatically available for the next job inside the warehouse;
- If there is a waiting period for a robot because of collision avoidance, the wait interval is constant and is the same as the time interval it takes a robot to move from one pickup point to the next. After each wait interval, the robot will recalculate and decide on whether to proceed or wait for another interval, to avoid a collision;
- Robot speed is assumed to be constant;
- A robot can only pick or bin one part at a time;
- No batching is allowed;
- Any robot can pick or bin any part, i.e. there are no restrictions on part size, weight etc.; and
- The model assumes the parts to be picked and binned are available immediately, all input data is of a deterministic nature, and the warehouse environment stays constant during the optimisation run.

4.2 Single objective algorithm framework

The algorithm framework developed for solving the single objective part sequencing and allocation problem includes determining the actual route of a robot through the warehouse. Thus, as part of the development of the algorithm framework the subproblem of routing the robot is solved.

The robot routing problem can be seen as a subproblem of the part sequencing and allocation problem. A suitable routing heuristic must be selected before it can be used in the part sequencing and allocation algorithm framework. The framework for solving the single objective part sequencing and allocation problem can be seen in Figure 10. The part sequencing and allocation algorithm framework uses the best routing strategy along with the four chosen metaheuristics to find the best possible solution for the sequencing and allocation problem.

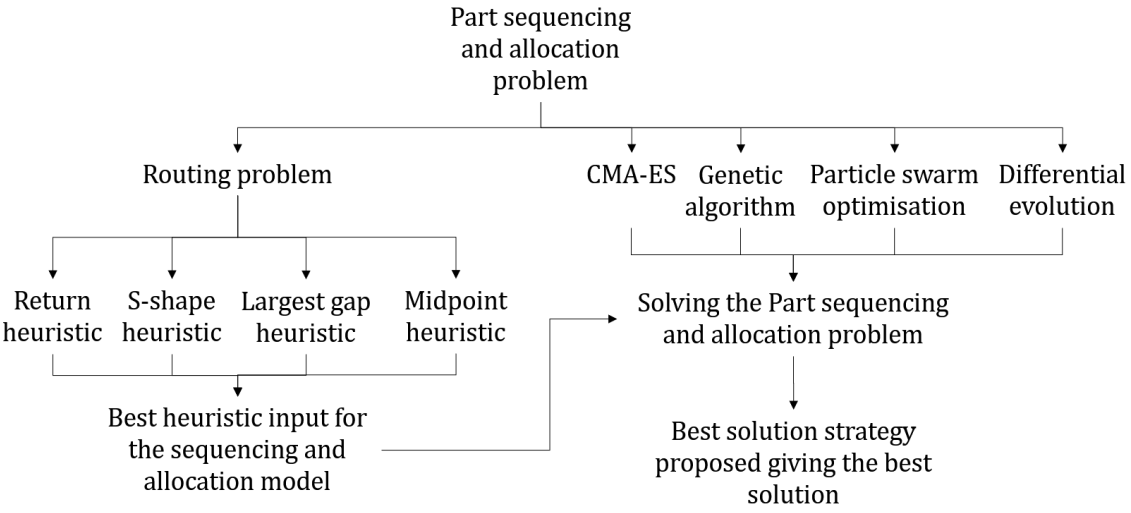

Figure 10: Solution strategies

The actual part picking and sequencing problem consists of a number of parts that need to be picked and binned by a number of robots in the quickest way possible. Given the sequence and allocation of parts from the metaheuristic, the path used for picking and binning is determined by the routing heuristic. The robots must complete all of their picking and binning tasks in the given sequence in the minimum amount of time. The picking and binning sequences are allocated in turn to respective robots. The picking and binning sequences and part allocations are optimised using the four metaheuristics discussed earlier; namely, the CMA-ES, GCPSO, SaNSDE, and GA. The algorithm framework developed to solve the part picking and sequencing problem is shown in Algorithm 14:

$I \triangleq$ Number of parts in the warehouse
$\boldsymbol{x} =$ Set of continuous variables given by the metaheuristic where $\boldsymbol{x} = 3 \times I$
$\boldsymbol{p} =$ Set of parts to be picked per robot
$\boldsymbol{b} =$ Set of parts to be binned per robot
$\boldsymbol{a} =$ Set of parts allocated to a robot

Algorithm 14: Single objective part allocation and sequencing algorithm

```
Initialise a warehouse with I number of storage spaces
Divide the n_x-dimensional candidate solution into 3 sets
for The first set → picking sequence do
    Arrange the first set in ascending order → part picking sequence (p)
end
for The second set → binning sequence do
    Arrange the second set in ascending order → part binning sequence (b)
end
for The third set → robot allocation do
    Given the number of robots (K) in the system complete the allocation
end
Assign each robot → next destination node
for All the parts in the warehouse do
    Route each robot to their destination using the routing heuristic
    if the next node is occupied then
        WAIT
    end
    else
        Move along the path
    end
end
if robot is done with picking and binning then
    state = DONE
end
else
    state = ACTIVE
end
if all robot states = DONE then
    return Fval = Total time for picking and binning
end
```

The algorithm framework initialises a warehouse with I number of parts and the metaheuristic is initialised with three times the number of parts. The continuous variables are then divided into three sets for the picking and binning sequences and the robot allocation. After the robot allocation each robot is assigned a next node. The robots are then routed to their next node. On the way they wait if a collision must be avoided. If a robot has picked and binned all its parts, then only can a robot be classified as complete. The algorithm stops to return the fitness value only if all the robots are classified as complete. The process flow of the algorithm can be seen in Figure 11.

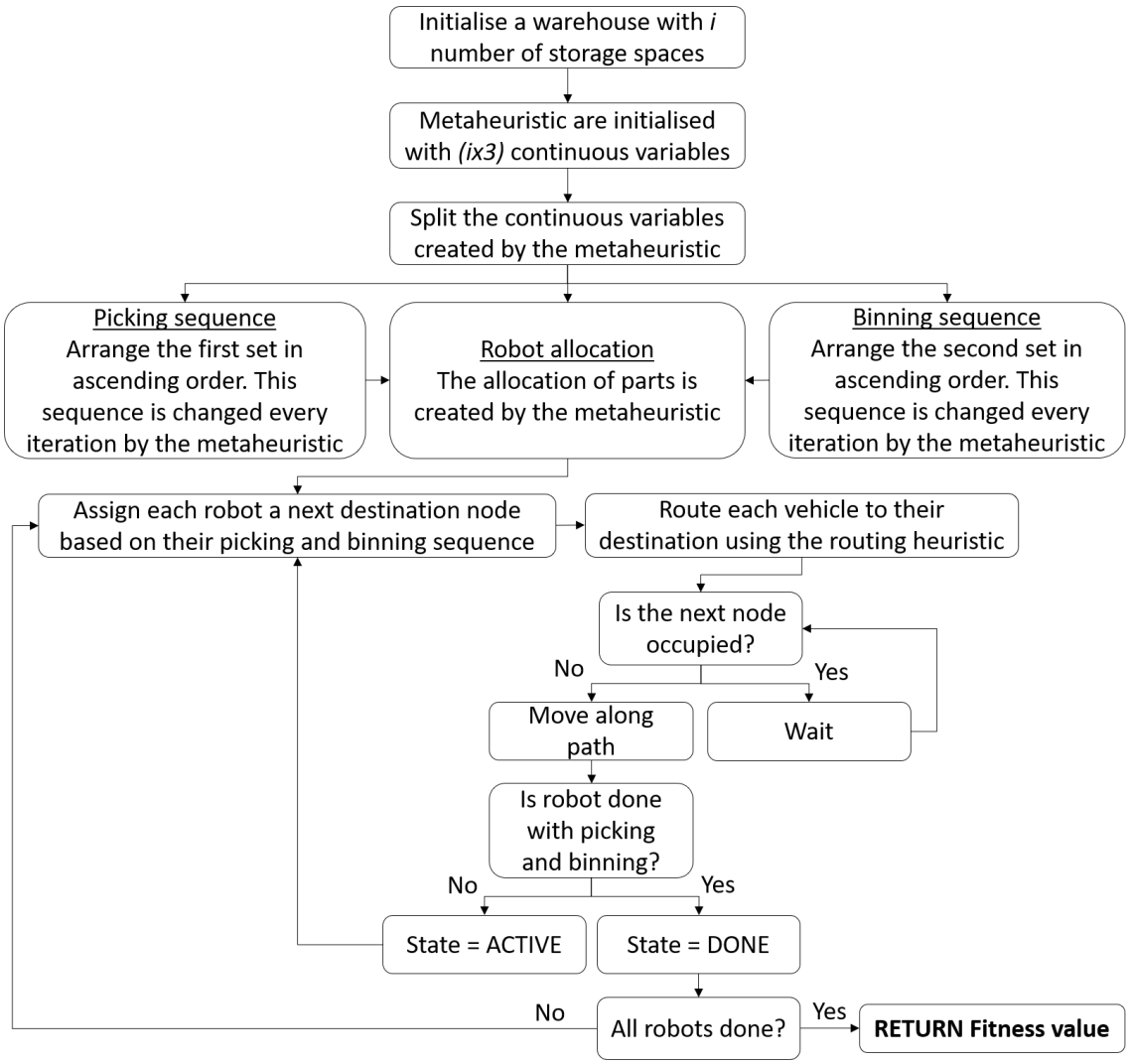

Figure 11: Process flow of algorithm

4.3 Summary

This chapter described the sequencing and allocation problem. The model was presented along with the objective function, the assumptions made, and the limitations of the model. The pseudocode and process flow of the algorithm framework were also presented in this chapter. The next chapter discusses the data analysis and the robot routing problem.

Chapter 5

Data analysis and the robot routing problem

This chapter describes the data analysis and the experimental setup for solving the robot routing problem. The results for the robot routing problem are also presented in this chapter. Section 5.1 describes the data for the parts and their locations within the warehouse. Section 5.2 discusses the robot routing problem, the robot routing algorithm, the experimental setup and Section 5.3 describes the results of the experiments. The robot routing is the first subproblem that is solved. The best routing heuristic from this chapter is later used to solve both the single and multi-objective part sequencing and allocation problem. Section 5.4 summarises this chapter.

5.1 Data description

The data used in this dissertation are the part location and part names as shown in Table 2. The data shown in Table 2 are used as input for the algorithm framework to know where each part is stored.

This chapter has been published in Croucamp and Grobler (2021).

Table 2: Part location ID and name

Location	Part name	Location	Part name
1	Presswasher	21	PIA 03
2	Nut M6	22	PIA 05
3	Washer	23	PIA 22
4	J-Nut	24	PIA 21
5	Nut 4.8X	25	PIA 1
6	Square Nuts	26	14N 139 AC
7	Screw	27	14026 AC
8	PIA 19	28	14N 139 BD
9	Screw 5X16	29	867 BD
10	Clip trim	30	Active
11	EB3B19952	31	Cruise RH
12	EB3B19953	32	Inner
13	541 BD	33	Chrome
14	PAD 885	34	Center
15	PAD 884	35	Bracket
16	Washer jet	36	Bracket LH
17	Scuff	37	Bracket RH
18	867 AD	38	Badge
19	PIA 06	39	Ext RR
20	PIA 20	40	Bumper

5.2 The robot routing problem

The four routing strategies considered are the return, s-shape, midpoint, and largest gap heuristics. The combined heuristic was not considered, given the fact that a robot is only allowed to pick or bin one part at a time, meaning that the combined heuristic will give the same result as the s-shape heuristic. These heuristics were discussed in detail in Section 2.1.1. These four heuristics were tested using the same sequence to pick and bin in the same warehouse layout.

The robot routing algorithms require the following data:

- The number of sequences to run. For the problem of routing a robot, 30 random sequences are generated for five different data sets; namely, 8 parts, 16 parts, 24 parts, 32 parts, and 40 parts. Each sequence was executed using all of the routing heuristics; namely, the return, s-shape, midpoint, and largest gap routing heuristics. All 30 random sequences for the five data sets were used to determine the best routing heuristic.
- The number of robots: a number between one and eight.
- The part name list as seen in Table 2. The part list is used to define the warehouse layout and determine the number of parts in the warehouse.
- The location ID of each part, which is used by the robots to know where each part is stored within the facility when binning or picking.
- The entrance gate and exit gate positions.

The pseudocode of the heuristic test procedure is shown in Algorithm 15.

Algorithm 15: Robot routing algorithm

```
Initialise a warehouse with I number of storage spaces per aisle and j number of aisles
Distribute parts evenly between the number of robots
for each heuristic do
    for each sequence do
        Pick and bin parts as per sequence
        Calculate time used
    end
    Return results
end
```

5.3 The robot routing results

The robot routing heuristics were tested with two robots. The results for the five data sets can be seen in Figures 12 to 16. From the figures, the results favour the return heuristic. The objective is to minimise the total time required for picking and binning. The return heuristic outperformed the other heuristics for all of the data sets tested.

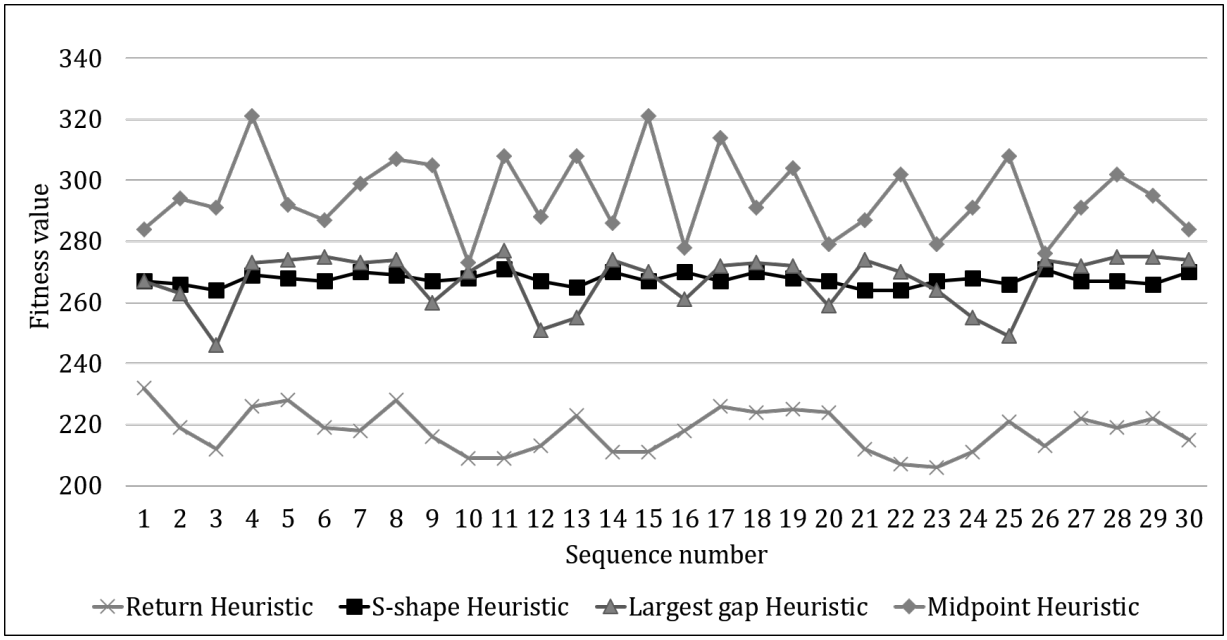

Figure 12: 8 parts routing results

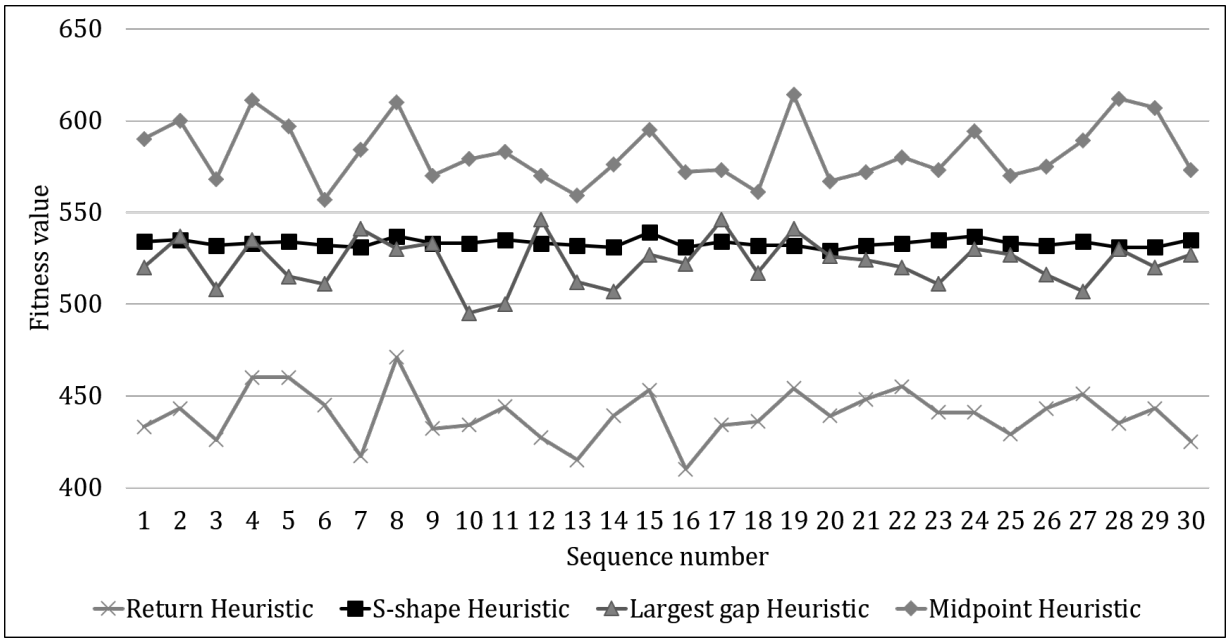

Figure 13: 16 parts routing results

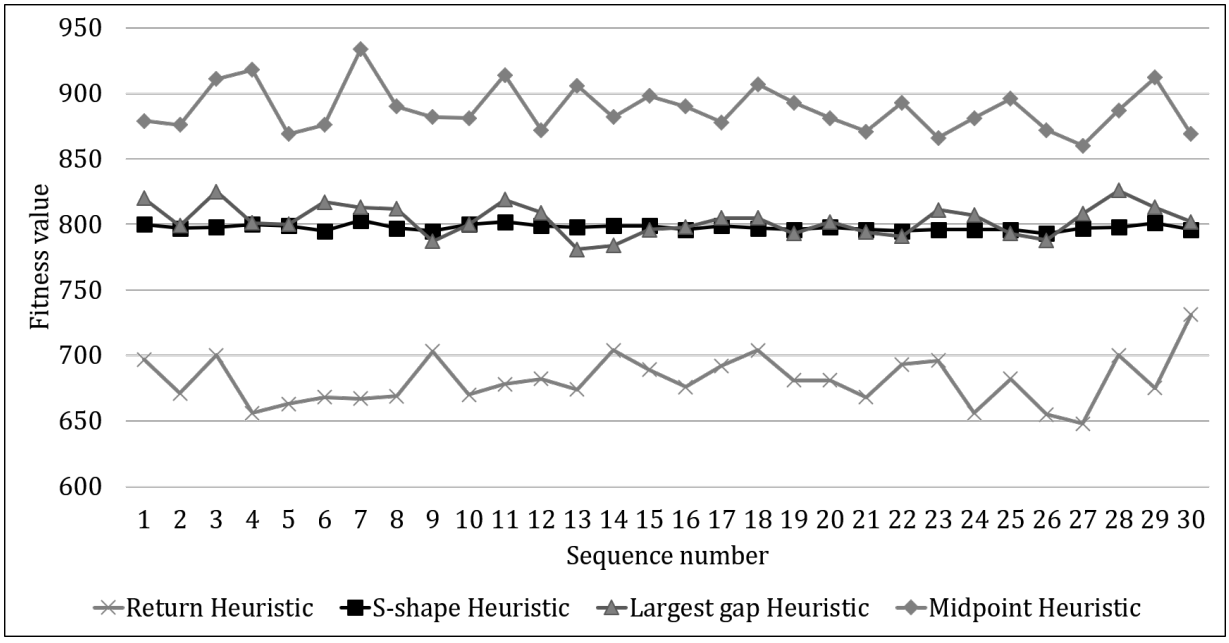

Figure 14: 24 parts routing results

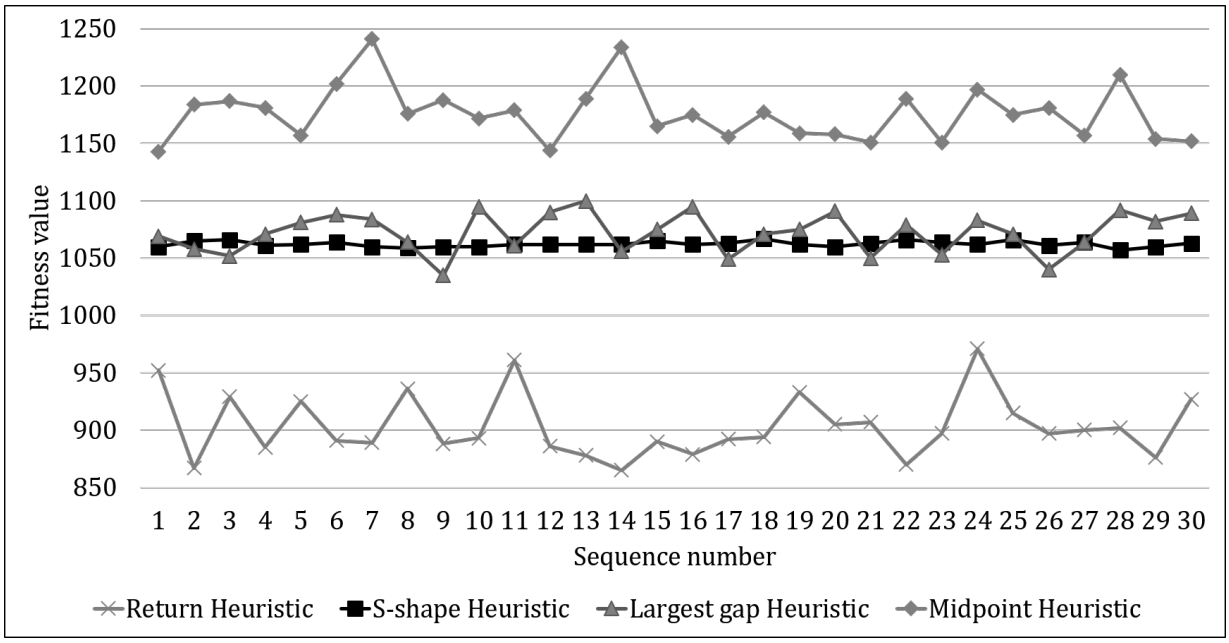

Figure 15: 32 parts routing results

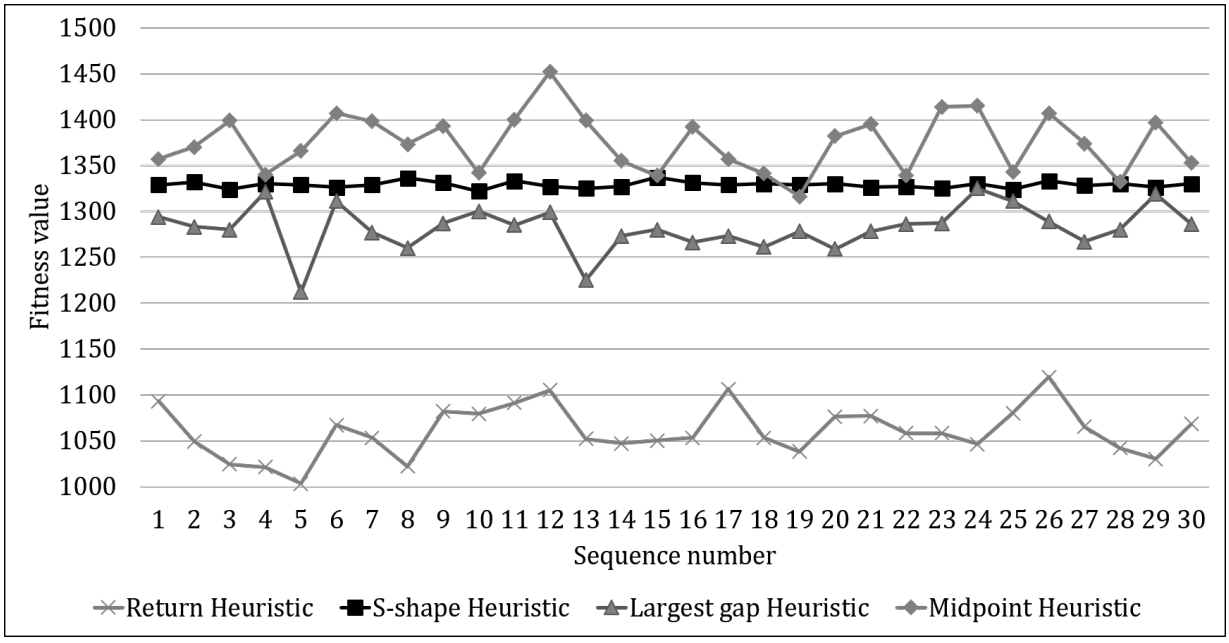

Figure 16: 40 parts routing results

The results for the four routing heuristics are summarised in Table 3. The table shows the average value obtained for all 30 random sequences, over all five data sets. The table also shows the standard deviation of the minimum values for the 30 sequences. From Table 3 it is clear that the return routing heuristic outperformed the others by a significant margin.

Table 3: Routing heuristics results

		Return heuristic	S-shape heuristic	Largest gap heuristic	Midpoint heuristic
8 Parts	**Average**	217.97	267.57	267.37	294.83
	Standard deviation	7.00	1.98	8.86	12.97
16 Parts	**Average**	439.43	533.17	522.70	582.70
	Standard deviation	13.88	2.12	13.11	16.74
24 Parts	**Average**	680.97	797.70	803.30	888.13
	Standard deviation	18.51	2.26	11.81	17.80
32 Parts	**Average**	903.33	1062.33	1072.10	1176.13
	Standard deviation	27.40	2.37	17.48	24.13
40 Parts	**Average**	1060.23	1328.83	1281.73	1374.90
	Standard deviation	27.31	3.41	24.66	31.63

The results of the statistical comparison in Table 4 were obtained by comparing the result of a heuristic's performance with each of the other heuristics' performance, for each data set. For every comparison, a Mann–Whitney U test at 5% significance was performed (using the two sets of 30 sequences of the two heuristics under comparison), and if the first heuristic outperformed the second heuristic statistically significantly, a win was recorded.

If no statistical difference could be observed, a draw was recorded. If the second heuristic outperformed the first heuristic, a loss was recorded for the first heuristic. The total number of wins, draws and losses was then recorded for all data sets of the heuristic under evaluation. As an example, (5-0-0) in row 1 column 2, indicates that the return heuristic significantly outperformed the s-shape heuristic 5 times over the 5 data sets. No draws and losses were recorded.

Table 4: Routing heuristic hypothesis testing results

	Return heuristic	S-shape heuristic	Largest gap heuristic	Midpoint heuristic	Total
Return heuristic	-	5-0-0	5-0-0	5-0-0	**15-0-0**
S-shape heuristic	0-0-5	-	2-1-2	5-0-0	**7-1-7**
Largest gap heuristic	0-0-5	2-1-2	-	5-0-0	**7-1-7**
Midpoint heuristic	0-0-5	0-0-5	0-0-5	-	**0-0-15**

From the hypothesis test it is clear that the return heuristic outperformed the other heuristics by a significant margin. From this point forward in the dissertation only the return heuristic will be considered for the routing of robots when solving the part sequencing and allocation problem.

5.4 Summary

This chapter discussed the robot routing problem in detail. The experimental setup and results were discussed and the best performing routing heuristic was determined statistically using the Mann–Whitney U test. The routing heuristic used for the rest of this document was concluded to be the return heuristic. The next chapter will discuss the part sequencing and allocation problem and the results obtained.

Chapter 6

Evaluating the single objective part sequencing and allocation algorithm

The single objective part sequencing and allocation problem requires the solution algorithm to assign all the parts to their respective robots, and sequence their binning and picking tasks. The part of the framework addressing the single objective part sequencing and allocation problem is discussed in Section 6.1. The algorithm parameters are discussed in Section 6.2. Sections 6.3 and 6.4 discuss the results for the single objective part sequencing and allocation problem. Section 6.5 investigates the specific performance of the GCPSO. Section 6.6 presents the hypothesis test results followed by a sensitivity analysis in Section 6.7. Section 6.8 summarises and concludes Chapter 6.

6.1 The part sequencing and allocation algorithm

The part sequencing and allocation problem was solved using the best performing routing heuristic from the robot routing part of the framework. The part sequencing and allocation problem consists of 40 parts, each part to be picked and binned. Each of the parts has to be assigned to a number of robots. Each robot is then responsible for the picking and binning of its assigned parts.

For each possible solution, the respective metaheuristic must create 120 continuous variables for the largest problem. The 120 variables are divided into three equal-sized sets of continuous variables. The three sets of continuous variables are used by the developed algorithm framework. The first set defines the sequence in which the picking takes place. The second set determines the sequence in which the binning takes place, and the third set is the allocation of parts to a robot in the system. The total number of variables in the system is then 120: 40 continuous variables to determine the picking sequence, 40 to determine the binning sequence, and 40 for the allocation of parts to the robots.

This chapter has been published in Croucamp and Grobler (2021).

The sequencing and allocation part of the algorithm framework requires the following data as input:

- The number of data sets;
- The data sets as an input;
- The number of robots allowed in the system as an input parameter, which is a number between one and seven;
- The part name list as seen in Table 2. The part list is used to define the warehouse layout;
- The location ID of each part. The location ID is used by the robots to identify where each part is stored within the facility when binning or picking;
- The metaheuristic parameters;
- The entrance gate and exit gate positions; and
- The maximum iteration count.

The sequencing and allocation part of the framework was tested using metaheuristics, namely the CMA-ES, GA, GCPSO, and SaNSDE. Each metaheuristic provides the fitness function with the number of continuous variables required for the problem size. The algorithm framework divides the continuous variables into the three sets of equal size as described earlier.

The first and second sets of continuous variables are sorted in ascending order. The index number of the lowest variable is used to identify the part that is associated with that index number. This process continues until both sets one and two have been sequenced successfully. For the third set the same process is followed but the list is divided by the number of robots available. Each robot will then receive the same number of parts to pick and bin.

Given that the parts are sequenced and allocated to robots, the algorithm framework calculates the fitness value of the sequences and allocation. The fitness value is returned to the metaheuristic for it to make an improvement. The metaheuristic run through the sequences until a near-optimal solution is found or the maximum iterations stopping criterion is reached.

The performance of the five metaheuristics was tested within the algorithm framework on five different problem sizes, derived from real-world data. Problem one has 8 parts to be sequenced and allocated. The second problem has 16 parts and the third has 24 parts. Problem number four has 32 parts to be sequenced and allocated and problem five has 40 parts to sequence and allocate.

For each problem size there are set parameters for all the metaheuristics. The population size remains constant over all of the metaheuristics and the different problem sizes. The population was set to 100 individuals.

The maximum iterations criterion was determined for each problem size. All metaheuristics were run as part of an empirical analysis on all five problem sizes. For each problem size the fitness values and diversity values were obtained. A population's diversity or solution space diversity (SSD) is defined as:

$$SSD = \frac{1}{n_s}\sum_{i=1}^{n_s}\sqrt{\sum_{j=1}^{n_x}(x_{ij}(t) - \bar{x}_j(t))^2} \tag{60}$$

where n_s denotes the population size and n_x is the number of variables in each individual. $x_{ij}(t)$ is the position of the j^{th} dimension of the i^{th} entity at time t, and $\bar{x}_j(t)$ is the mean of the j^{th} dimension of all the particles at time t (Vesterstrom et al., 2002). Given the fitness values and the diversity values, the maximum iterations needed for each problem size were determined. The point of no further improvement was identified using the plot of fitness function value versus the number of iterations. Given these results the maximum number of iterations was determined for each problem size respectively.

The problem size of 8 parts and the problem size of 16 parts had the maximum iterations set to 500. For the 24 parts problem the maximum number of iterations was increased to 750. The problem solving 32 parts had the iteration maximum parameter set to 900 and for the biggest problem size, 40 parts, the maximum number of iterations was increased to 1250.

6.2 Single objective algorithm parameters

The metaheuristic parameters are shown in Table 5. The notation $x \to y$ is used to indicate that the associated parameter is decreasing or increasing linearly from x to y over 95% of the maximum number of iterations.

Table 5: Metaheuristic parameters

Parameter	Value used
General parameters	
Population size (constant for all metaheuristics	100
Max iterations (Size = 8) parts	500
Max iterations (Size = 16) parts	500
Max iterations (Size = 24) parts	750
Max iterations (Size = 32) parts	900
Max iterations (Size = 40) parts	1250
Number of independent simulation runs per problem size	30
GCPSO parameters	
Acceleration constant (c_1)	$2.0 \to 0.7$
Acceleration constant (c_2)	$0.7 \to 2.0$
Inertia weight	$0.9 \to 0.4$
SaNSDE parameters	
Probability of reproduction	$0.75 \to 0.25$
Scaling factor	$0.75 \to 0.125$
GA parameters	
Probability of crossover	$0.6 \to 0.4$
Probability of mutation	0.1
Blend crossover parameter	0.5
CMA-ES parameters	As specified in Auger and Hansen (2005).

6.3 Algorithm framework evaluation results

The experimental runs were completed using the parameters described. The results presented in this section include the fitness values for the median run of each metaheuristic, as well as the respective diversity values for the median run. The algorithm framework was able to generate feasible and usable solutions. The results for all the metaheuristics are shown by problem size. The figures present the median run (fitness function over time) of thirty runs for the problem size. Figure 17 shows the results for the sequencing and allocation of 8 parts. Figure 18 shows the results for 16 parts, Figure 19 shows the results for 24 parts, Figure 20 shows the results for 32 parts and Figure 21 shows the results for the largest problem of 40 parts to be sequenced and allocated. The fitness value was calculated as the total time it takes to complete all of the picking and binning processes. The fitness value represents the number of time intervals it takes to complete the picking and binning processes.

The best performing metaheuristic for the sequencing and allocation of 8 parts on the median run was the SaNSDE followed by the GCPSO, CMA-ES and the GA, in that order.

The SaNSDE found the best solution with a fitness value of 185, the GCPSO returned a value of 187, and both the GA and CMA-ES found the best fitness value to be 188. The results for sequencing and allocating 16 parts (Shown in Figure 18), show that the CMA-ES, GA, and SaNSDE found the same minimum value of 375. The GCPSO was not able to find the same minimum value and ended at a fitness value of 379.

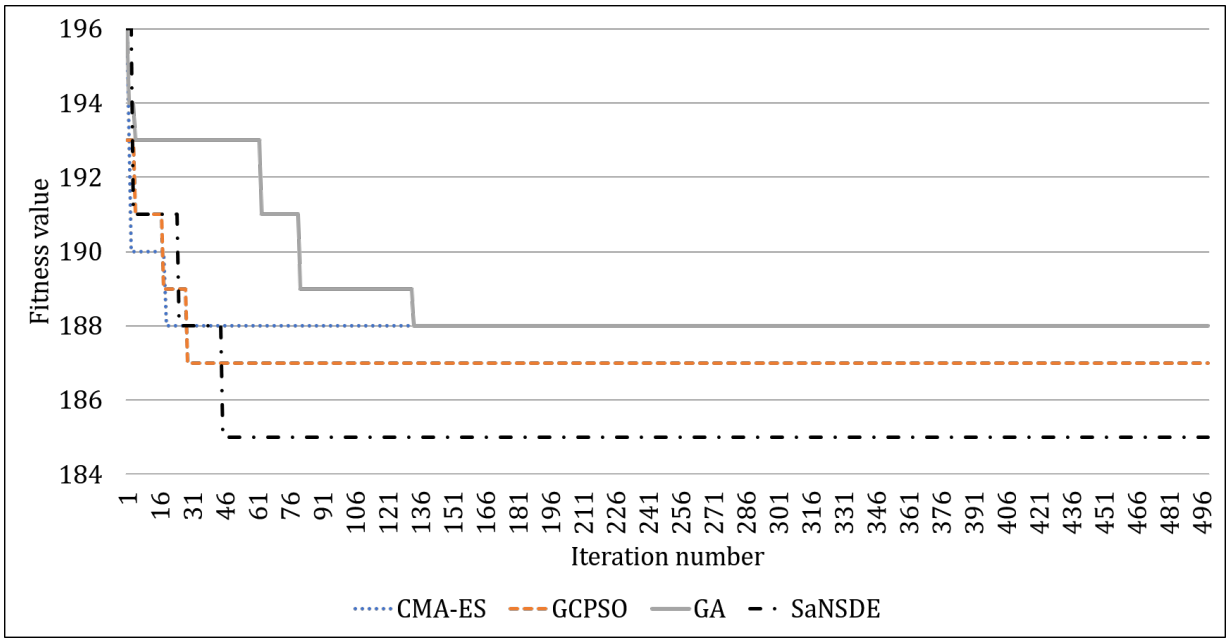

Figure 17: 8 parts fitness value results

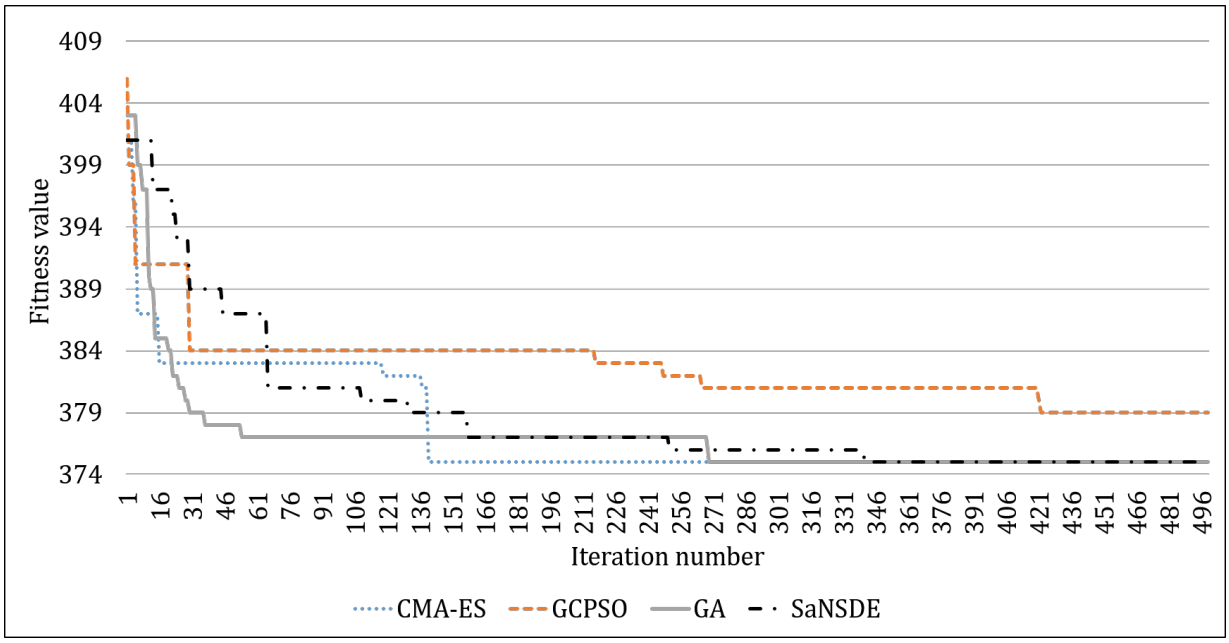

Figure 18: 16 parts fitness value results

The results for the sequencing and allocation problem with 24 parts (shown in Figure 19), shows the best performing metaheuristic for the sequencing and allocation of 24 parts to be the SaNSDE with a fitness value of 592, followed by the CMA-ES slightly behind, with a fitness value of 593, The GA was third with a fitness value of 595, while the GCPSO performed the worst with a fitness value of 600.

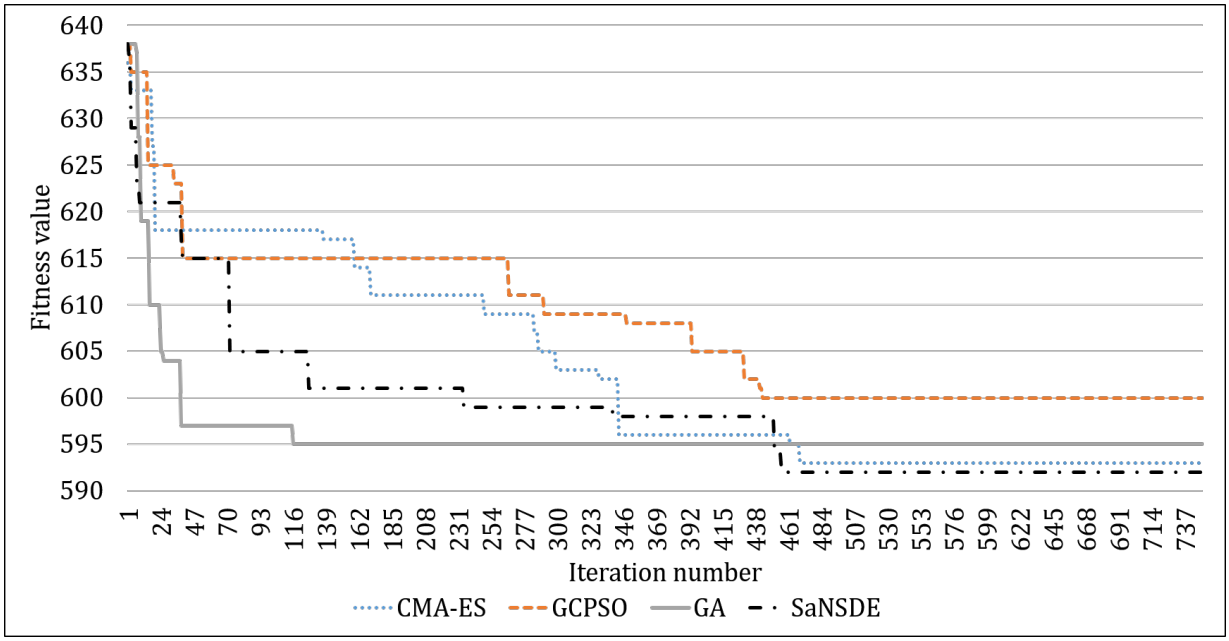

Figure 19: 24 parts fitness value results

The results in Figure 20 show the best performing metaheuristic for sequencing and allocation of 32 parts to be the CMA-ES. The CMA-ES outperformed the GA, SaNSDE, and the GCPSO with a fitness value of 784 compared to a fitness value of 788 for the GA, 789 for the SaNSDE, and a fitness value of 803 for the GCPSO.

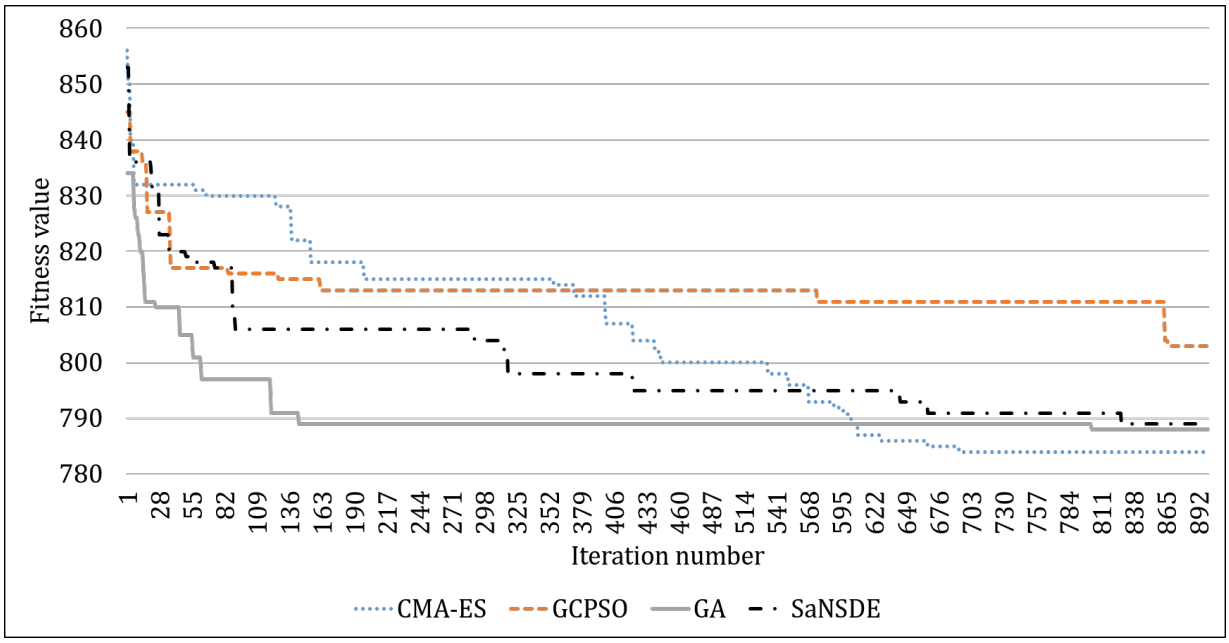

Figure 20: 32 parts fitness value results

The results for the largest problem of 40 parts are shown in Figure 21. The metaheuristic that showed the best performance in finding the lowest fitness value was the CMA-ES. The CMA-ES was able to find a feasible solution with a fitness value of 929. Both the SaNSDE and the GA performed equally well with a fitness value of 936. The worst performing algorithm was the GCPSO which found its lowest fitness value to be 965.

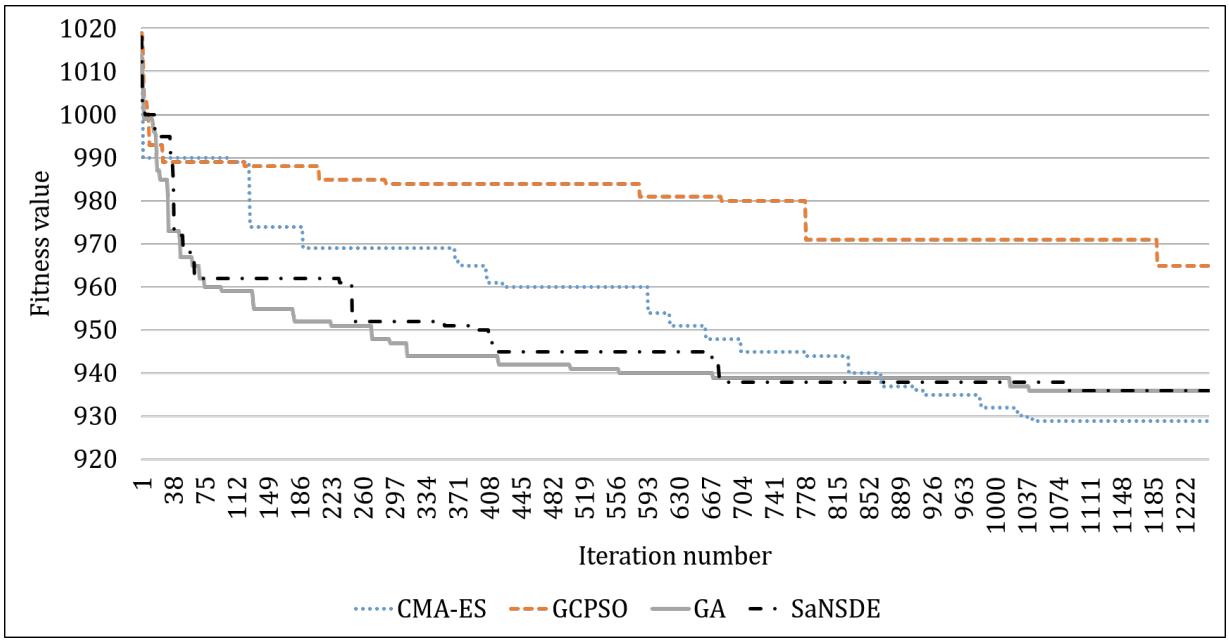

Figure 21: 40 parts fitness value results

The results for the five data sets discussed are summarised in Table 6. The table shows the average minimum value obtained for all 30 runs for each data set. The table also shows the standard deviation of the minimum values for the 30 runs.

Table 6: Metaheuristics results

		CMA-ES	GCPSO	GA	SaNSDE
8 Parts	**Average**	187.37	**187.03**	**187.03**	186.23
	Standard deviation	1.67	1.52	1.96	1.45
16 Parts	**Average**	**374.97**	378.27	375.27	375.20
	Standard deviation	2.58	3.12	2.55	1.63
24 Parts	**Average**	593.33	602.67	595.60	**591.73**
	Standard deviation	3.28	6.33	4.88	1.95
32 Parts	**Average**	**783.47**	803.93	787.70	788.43
	Standard deviation	3.88	7.88	5.18	3.06
40 Parts	**Average**	**929.53**	965.37	935.83	936.17
	Standard deviation	7.99	7.52	5.41	3.12

6.4 Single objective diversity function results

This section investigates the solution space diversity of each algorithm on each problem. The population diversity is an indication of the spread of the population through the search space. If the diversity value increases it means that the population is spread wider and the search area has been broadened and the algorithm is exploring the search space. When the diversity value decreases it indicates that the search area has been reduced and the algorithm is further exploiting good solutions. The diversity results in this section are used to verify that the metaheuristics do not prematurely converge to local optimal solutions, but continue exploring the search space throughout the optimisation process. The diversity values presented in Figures 22 to 26 were calculated using Equation 60. The diversity results for all five data sets follow the same pattern for each metaheuristic. The CMA-ES shows a low diversity value compared to the other metaheuristics. The diversity value of the CMA-ES grows at a low linear rate over the duration of the iterations. The CMA-ES's diversity values show that it does gradually explore more as the optimisation process progresses.

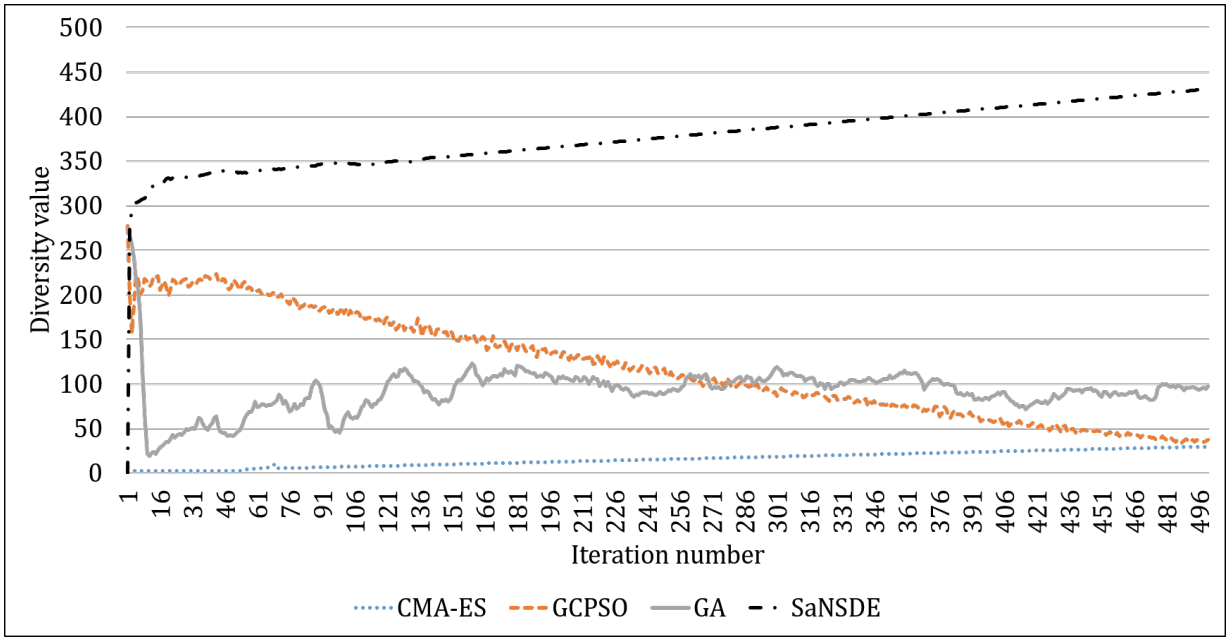

Figure 22: 8 parts diversity value results

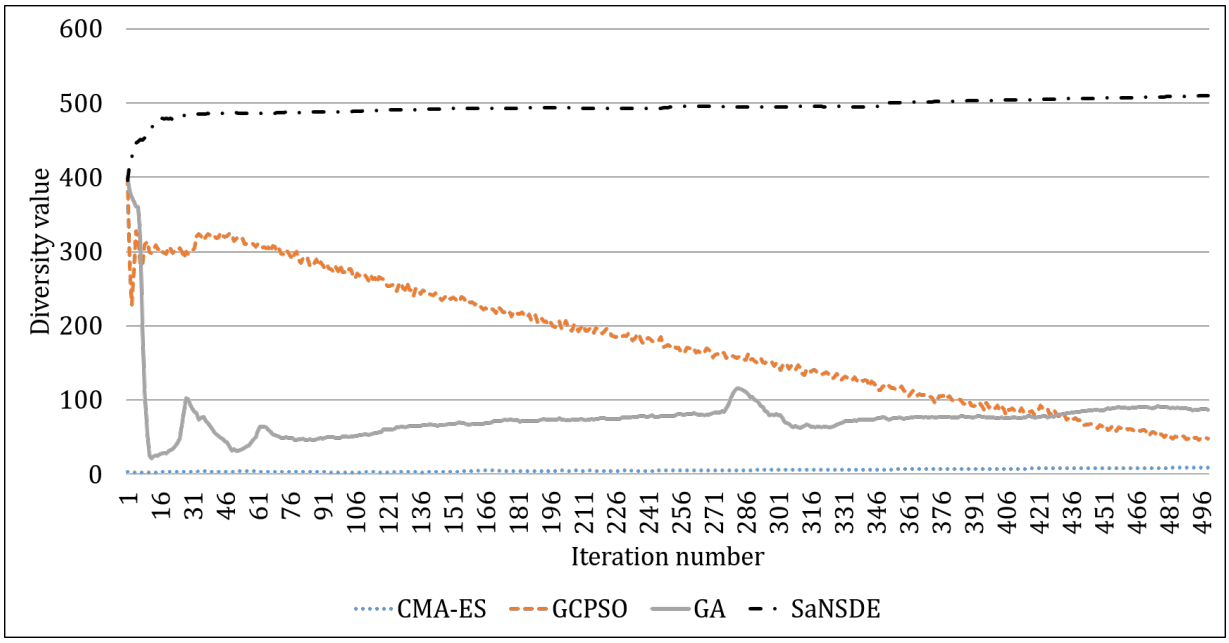

Figure 23: 16 parts diversity value results

The diversity profile for the GCPSO is ideal for an optimisation algorithm. The diversity of the GCPSO starts off high and then gradually decreases over time, resulting in a desirable balance between exploration and exploitation.

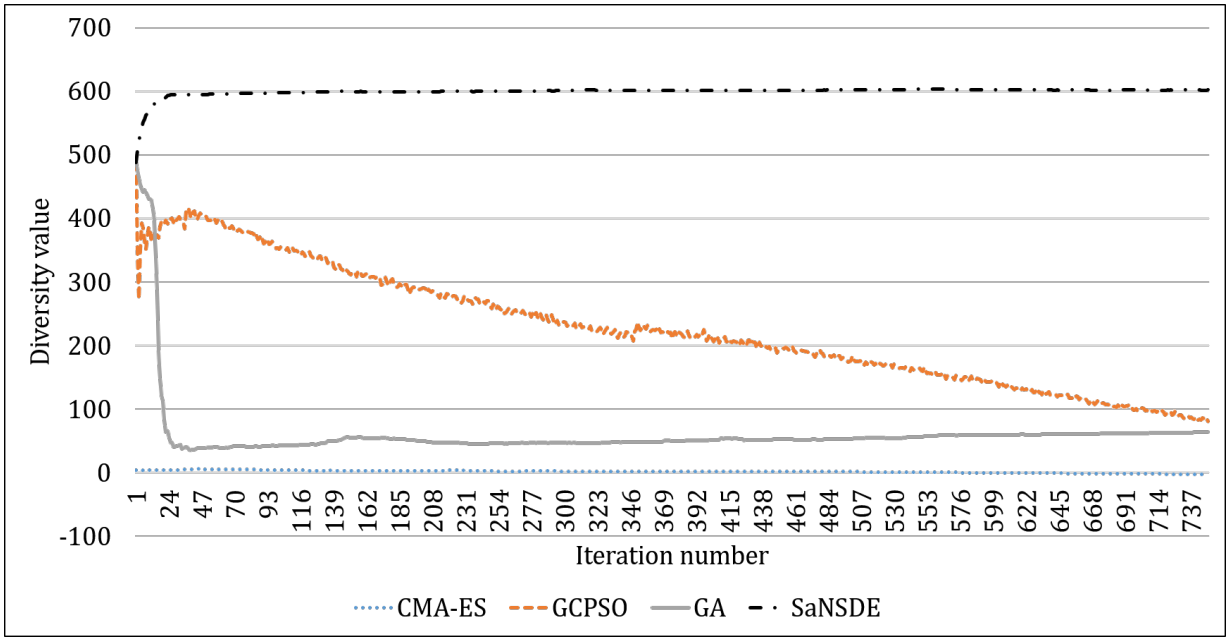

Figure 24: 24 parts diversity value results

The GA exhibits a relatively high diversity value which decreases quickly. Thereafter, the diversity remains quite stable with a slight increase towards the end of the algorithm's run.

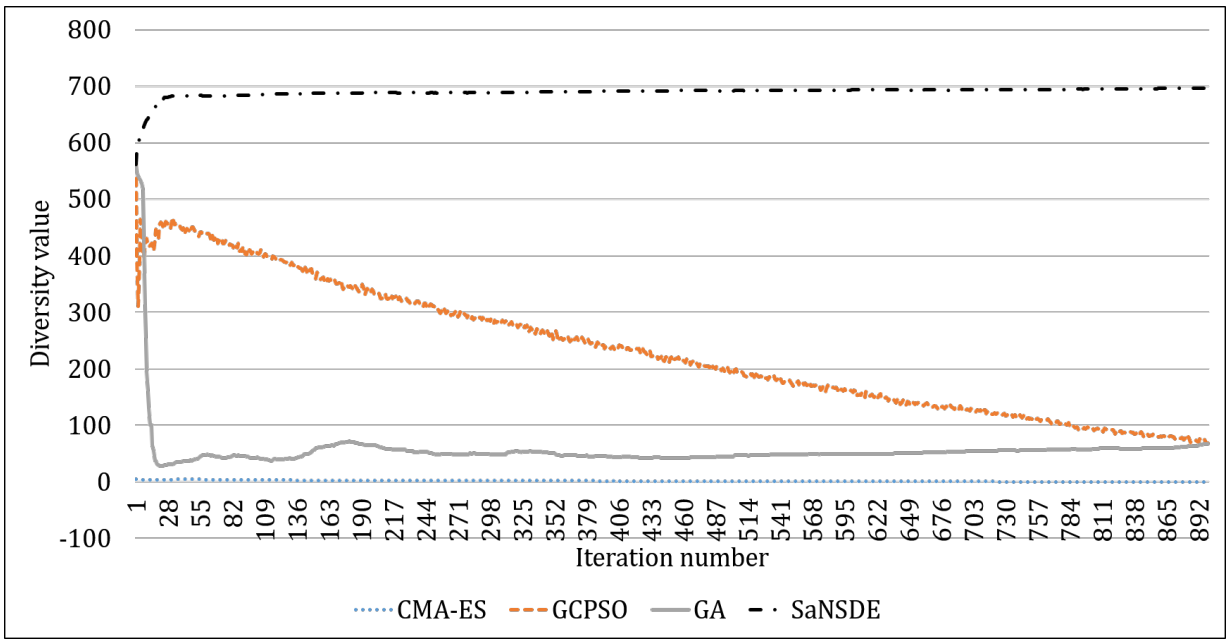

Figure 25: 32 parts diversity value results

The profile followed by the diversity values of the SaNSDE is in contrast to the rest of the metaheuristics in the sense that it starts at a high value and then increases sharply to an even higher value. After only a couple of iterations the diversity is very high.

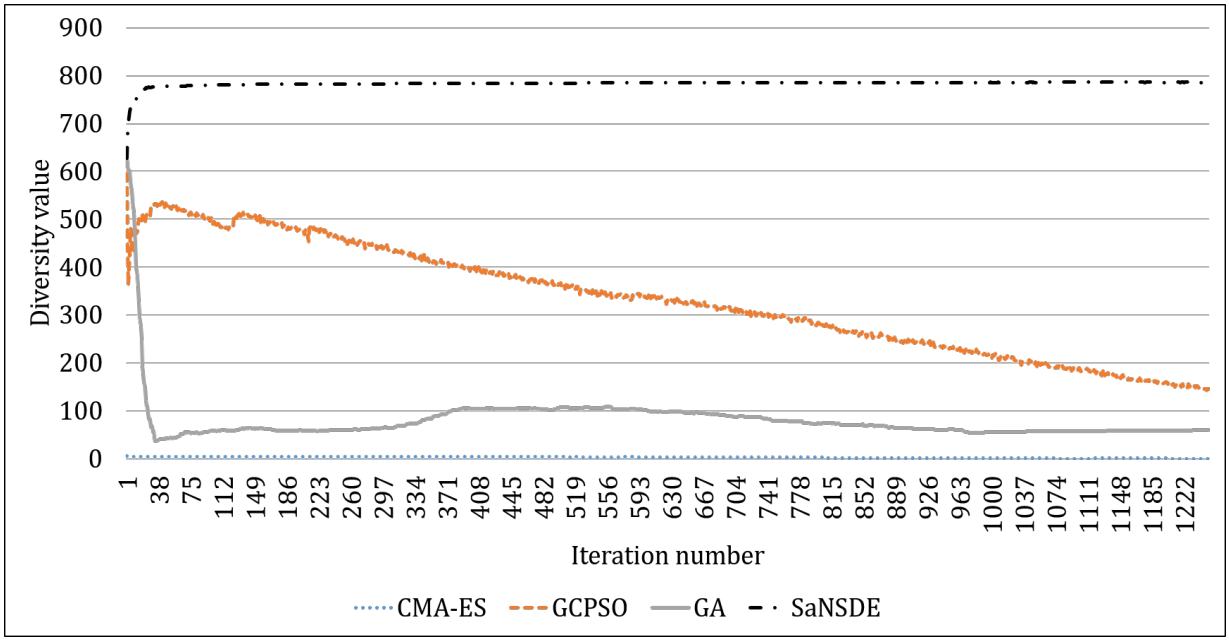

Figure 26: 40 parts diversity value results

Although all of the metaheuristics follow similar patterns for each data set, the diversity values increase as the problem size increases. For the problem of 8 parts the diversity values range from 0 to 450, whereas with the problem of 40 parts, the diversity values range between 0 and 800. It is positive to note that for all the metaheuristics over all the data sets the diversity value never decreases to zero. This result shows that none of the algorithms have prematurely converged to a local optimal solution. The required diversity in a problem is like

algorithm performance; it is problem-specific and can change due to problem size and problem complexity.

6.5 Single objective metaheuristic results analysis

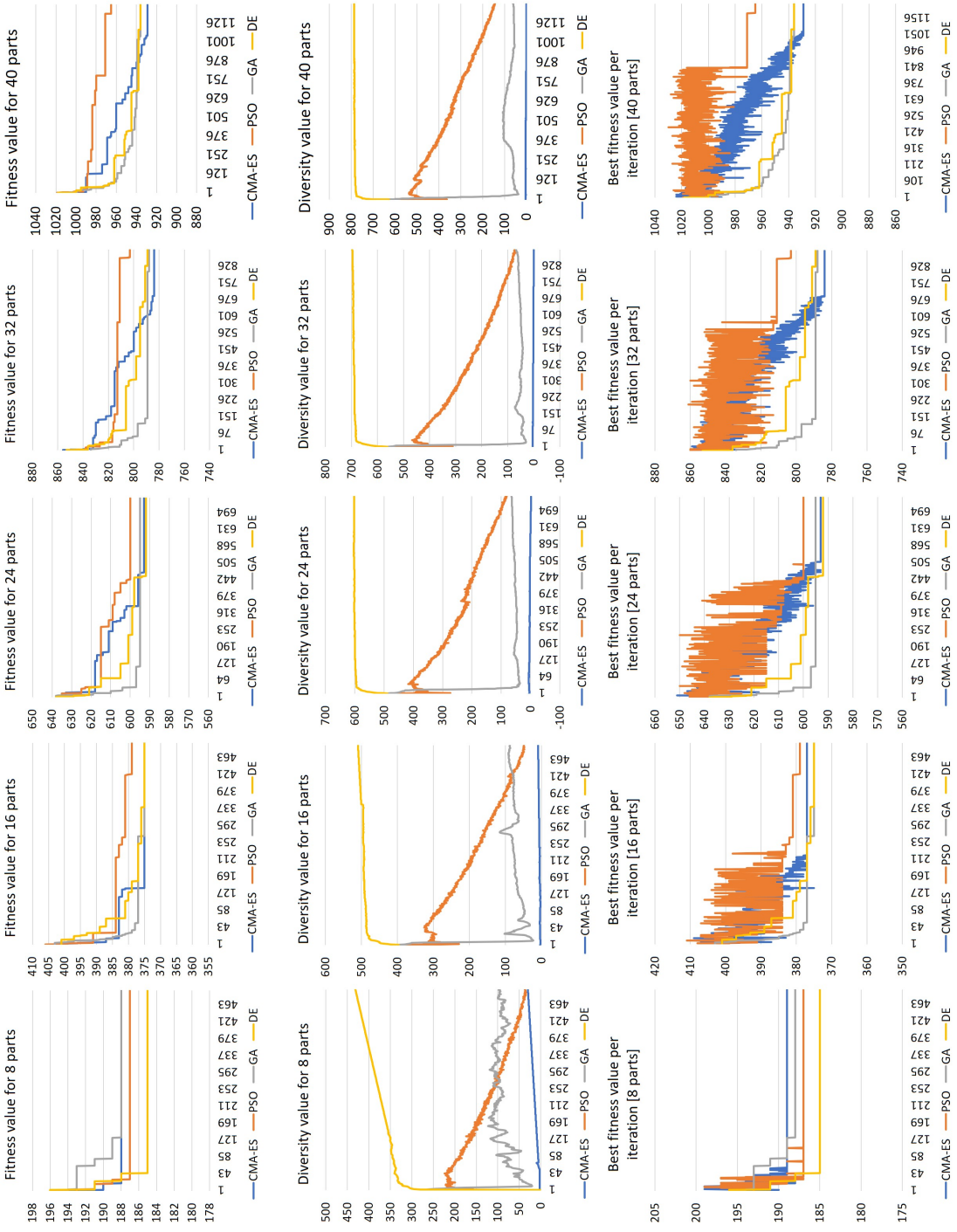

Figure 27: Single objective metaheuristic results

The first five graphs in the top row of Figure 27 show the overall best-found fitness value versus iterations for the 8, 16, 24, 32, and 40 part problem. The middle row of graphs in Figure 27 show the diversity of the solution versus iterations and the bottom row of graphs in Figure 27 show the best-found solution per iteration versus iterations for the 8, 16, 24, 32, and 40 part

problems. Interpreting the results, the same algorithm behaviour can be seen for the 16, 24, 32, and 40 part problem: Significantly worse performance can be seen from the GCPSO algorithm.

It is key to note the nature of each metaheuristic tested for the single objective part sequencing and allocation model. The SaNSDE and GA has similar characteristics when considering the generation of the next population. The SaNSDE and GA will keep the best solution found in the current population (elitism) and include that best solution in the next generation, ensuring that the best fitness value in the next generation will be either better or equally as good as the current generation. This characteristic is not the same for the GCPSO and CMA-ES. Although the CMA-ES and GCPSO keeps memory of the best-found solutions over the iterations, the CMA-ES and the GCPSO does not include the best solution automatically into the next generation. This characteristic may have given the advantage to the SaNSDE and GA over the GCPSO. Although the CMA-ES does not share this characteristic, the CMA-ES was still able to outperform the SaNSDE and GA due to maintaining a smaller population diversity to better exploit solutions.

The implementation of linearly adjusting the metaheuristics parameters and specifically the GCPOS's parameters c_1, c_2 and inertia weight gave it a theoretical "ideal" diversity profile i.e. linearly decreasing from an exploration phase into an exploitation phase. The SaNSDE had the highest diversity overall which in combination with the characteristic of keeping the best solution, might have benefited the SaNSDE, exploring larger areas (because of high diversity) but still having the knowledge to know where the last best solution is.

Considering the best fitness values obtained over the number of iterations, the SaNSDE and the GA performed well and continually improved on their best solution as the iterations continued. Due to the characteristic of keeping the previous best solution in the next population the fitness value decreases in what looks like a step wise manor. The CMA-ES had a much smaller range in terms of fitness function values over time, compared to the GCPSO which led to the assumption that the smaller diversity increased the efficiency of the search. Given the best fitness values obtained per iteration the GCPSO was doing more exploration over the period of the run than the other metaheuristics (the range of fitness values is much larger than expected and what is shown by the CMA-ES), however near the end of the run the GCPSO abruptly stops exploring and rapidly converged to a single solution rather than systematically exploiting good solutions. It is important to note, however, that the GCPSO did converge to a solution and that the convergence was not premature.

6.5.1 Investigating the solutions found by the GCPSO

In order to understand the solutions found by the GCPSO all the fitness value results over one run were analysed. The fitness values were obtained and converted into a histogram to analyse. The histogram is shown in Figure 28. The histogram shows the frequency that the GCPSO has found each of the solutions as seen in the median run.

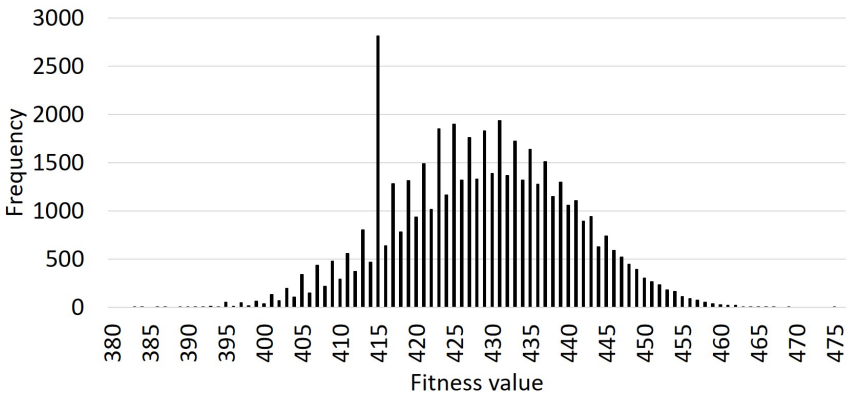

Figure 28: Histogram of fitness values obtained (16 Part problem, GCPSO)

The histogram shows that the GCPSO was exploring the same solutions multiple times during a run. It is clear that the GCPSO found the same fitness value (415), 2816 times during the run. The result indicates that in the solution space there are a large number of candidates that maps to the same part allocation and sequence. This observation can be attributed to the mapping mechanism used to convert the continuous variables to the objective space, which will have an impact on the efficiency of search. This behaviour could explain the poor performance of the GCPSO.

6.5.2 Analysis into the GCPSO's parameter performance

A number of further experiments were conducted to test the sensitivity of the population size and main parameters of the GCPSO, versus the originally used parameters of a population size of 100 and parameters c_1, c_2, and w linearly adjusting over time. The experiments included running the GCPSO with:

- A smaller population size: 50
- A larger population size: 150
- An even larger population size: 200, and
- Keeping the acceleration and inertia weight parameters constant ($c_1 = 2.0$, $c_2 = 0.7$, $w = 0.9$)

From Figure 29 there is no clear evidence of the GCPSO improving due to the increase or decrease in population size. Also, when considering a constant c_1, c_2, and w value the algorithm maintains a high diversity throughout the optimisation run and does not converge to as good a solution as obtained with the original parameters. These findings show that the GCPSO is insensitive to the population size parameter and that higher diversities do not improve performance.

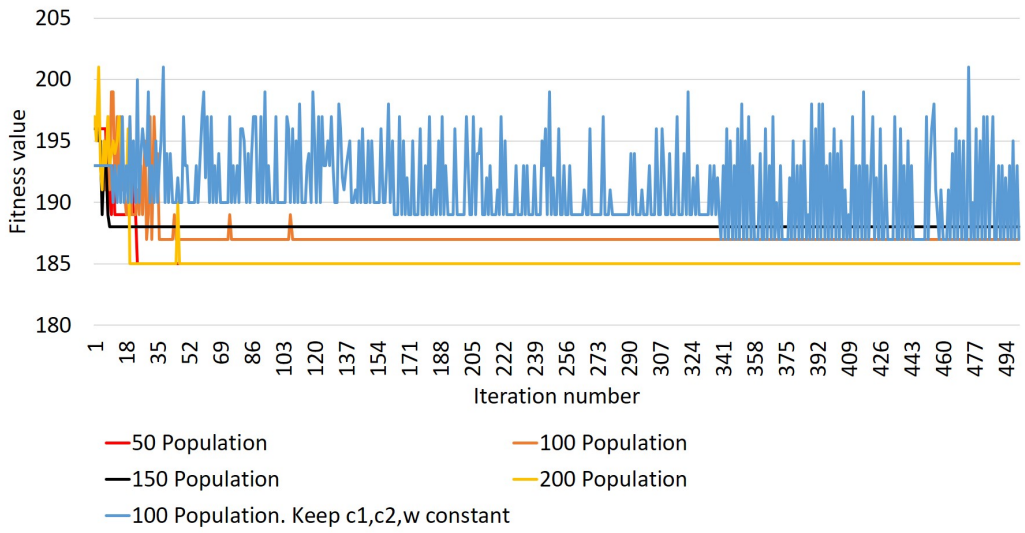

Figure 29: Analysis into the GCPSO's parameter performance

The next section discusses the results of the hypothesis tests performed on the results. The hypothesis test analyses the statistical significance of the results obtained by each metaheuristic.

6.6 Single objective hypothesis test results

The results of the statistical comparison in Table 7 were obtained by comparing the result of a metaheuristic's performance with each of the other metaheuristics' performances, for each data set. As described earlier in the dissertation, for every comparison a Mann–Whitney U test at 5% significance was performed (using the two sets of 30 sequences of the two heuristics under comparison) and if the first metaheuristic statistically significantly outperformed the second metaheuristic, a win was recorded. If no statistical difference could be observed, a draw was recorded. If the second metaheuristic outperformed the first metaheuristic, a loss was recorded for the first metaheuristic.

The total number of wins, draws and losses were then recorded for all data sets of the metaheuristic under evaluation. As an example, (4-1-0) in row 1 column 2, indicates that the CMA-ES significantly outperformed the GCPSO algorithm four times over the five data sets. One draw and no losses were recorded.

Table 7: Hypothesis testing results

	CMA-ES	**GCPSO**	**SaNSDE**	**GA**	**Total**
CMA-ES	-	4-1-0	2-2-1	2-3-0	**8-6-1**
GCPSO	0-1-4	-	0-1-4	0-1-4	**0-3-12**
SaNSDE	1-2-2	4-1-0	-	1-4-0	**6-7-2**
GA	0-3-2	4-1-0	0-4-1	-	**4-8-3**

Given the results from the hypothesis test, the best performing metaheuristic can be confirmed. The results from the hypothesis test conclude that the CMA-ES has outperformed the other metaheuristics on the sequencing and allocation problem with the greatest number of wins. The CMA-ES algorithm had a total of eight wins, six draws, and one loss. The metaheuristic that performed secondbest was the SaNSDE with six wins, seven draws, and two losses.

The GA was the third-best metaheuristic and had a total of four wins, eight draws, and three losses. The worst performing metaheuristic was the GCPSO with zero recorded wins, three draws, and twelve losses. From here onward the CMA-ES algorithm is the recommended algorithm for use in the sequencing and allocation algorithm framework and it is used to evaluate the sensitivity of the solution. The next section investigates the sensitivity of the model.

6.7 Single objective algorithm framework sensitivity analysis

The purpose of this section is to determine how the framework will behave with a change in input parameters. As discussed earlier, there are a couple of input parameters that can be changed to have an impact on the solution.

The sensitivity analysis focused on the increase in the number of robots in the warehouse and the effect this might have on the fitness function. The sensitivity analysis was conducted using the same experimental setup as described earlier in this chapter.

The number of robots was increased from two (original setting), to a number where no significant improvement could be identified. The maximum number of robots identified was seven. The algorithm was run five times for each of the number of robots considered and the results of the median run were used to compare the scenarios. The results of the sensitivity analysis can be seen in Figure 30.

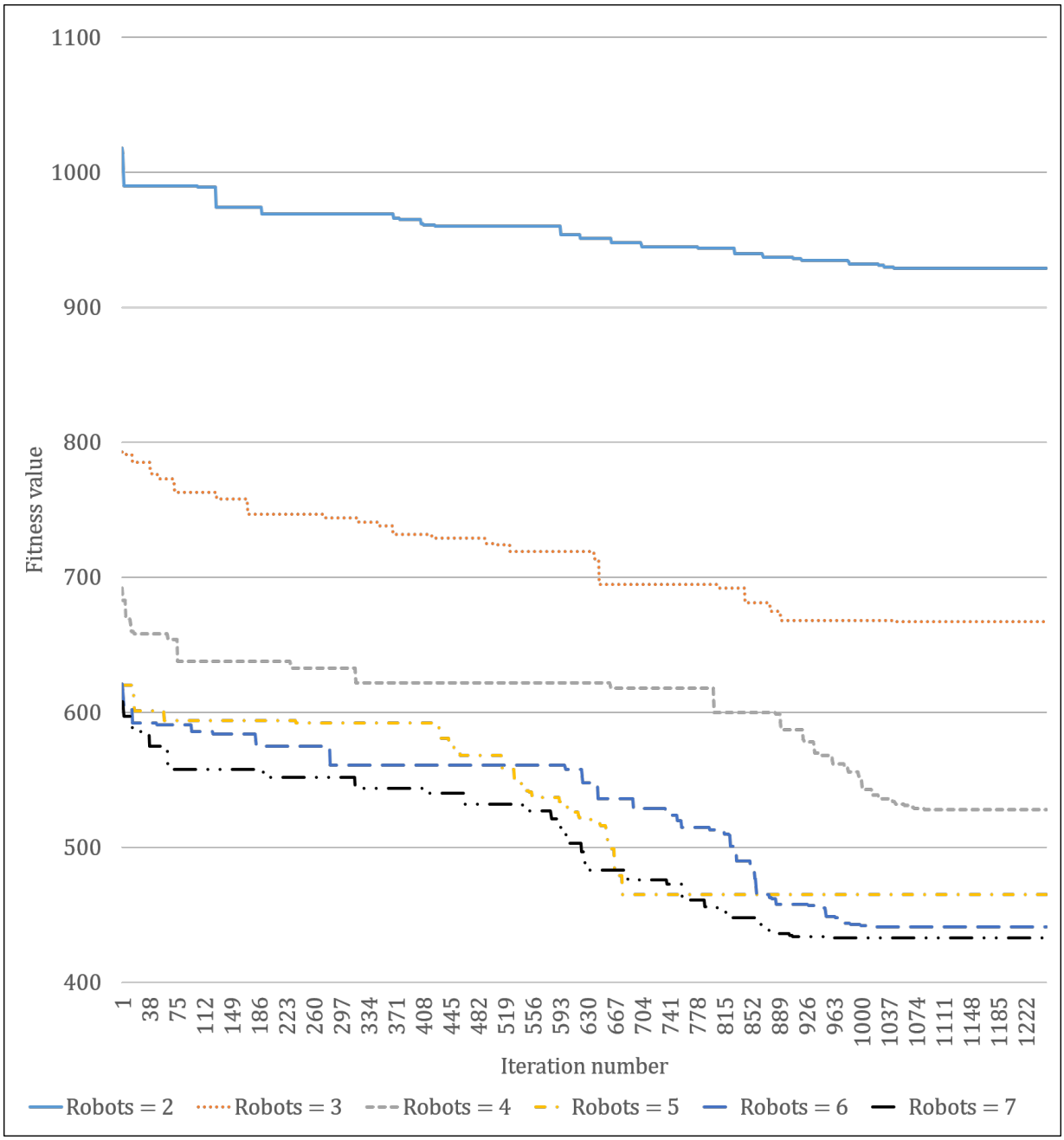

Figure 30: Sensitivity analysis

The benefit of adding more and more robots became smaller and smaller as the number of robots increased. When increasing the number of robots from two robots to three, an improvement of 28.2% could be seen on the fitness value. The improvement from three robots to four showed an improvement of 20.8%. The percentage improvement decreased further to 11.9% when introducing a fifth robot. Introducing another robot, making a total of six, improved the fitness value by only 5.2%. The maximum number of seven robots showed a slight improvement of 1.8%. The maximum number of robots is reached because the workload remains constant. The robots are eventually able to handle the constant workload, therefore it is not cost-effective to add any more robots.

6.8 Summary

The aim of this chapter was to evaluate the performance of the algorithm framework through the application of CMA-ES, GCPSO, GA and SaNSDE as metaheuristics in the part sequencing and allocation algorithm framework. The four metaheuristics were applied in the sequencing and allocation algorithm framework as discussed in Chapter 4. The CMA-ES outperformed the other metaheuristics on the five data sets tested. The CMA-ES, which was the best performing metaheuristic, was also used in the sensitivity analysis to determine the effect of the number of robots in the system. The sensitivity analysis showed that there is improvement opportunity by adding robots to the system.

Chapter 7

Data exploration for multi-objective correlation

The aim of this chapter is to identify different objective functions that can be considered for the multi-objective part sequencing and allocation problem and determining if there are any correlations between the identified objective functions. The results from the single objective part sequencing and allocation problem are used to explore and analyse the possible correlations between the identified alternative objective functions.

7.1 Objective functions considered for the multi-objective problem

A total of ten objectives can be minimised for the part sequencing and allocation problem. These objectives were investigated to determine which two were best suited for the multi-objective part sequencing and allocation problem discussed in the following chapters. The objectives that were considered include:

- Makespan: Makespan refers to the total time that elapses between the start and completion of all binning and picking tasks.
- Maximum idle time: Maximum idle time is the maximum time any robot is idle. Idle time is the time a robot is waiting for other robots to complete their binning and picking tasks.
- The sum of the idle time: The sum of the idle time is the sum of all the robots' idle time.
- Average idle time: Average idle time is the average time that all the robots have been idling.

This chapter was submitted to IEEE *Access*, Under review.

- Maximum waiting time: Maximum waiting time is the maximum time until any robot has to stop and wait to avoid collisions. A robot has to wait until it is possible to perform a task that has been allocated to it. The reason robots have to wait to do certain tasks is to avoid collisions with other robots since the aisles in the warehouse are not wide enough for two robots to pass each other. In the example waiting times shown in Table 8, robot two has the maximum waiting time equal to 35 minutes.
- Sum of the total waiting time: The sum of the total waiting time is simply the sum of each robot's total waiting time over the duration of the picking and binning time. In Table 8 it is the sum of all the values in the bottom row which adds up to 147 minutes.

Table 8: Example of total waiting times for three robots (Kleyn, 2020)

	Picking robot 1	Picking robot 2	Picking robot 3
	5	8	1
	6	6	1
	9	20	1
	8	23	6
	15	35	3
Total	43	92	12

- Average waiting time: Average waiting time is the average time for which all robots have to stop and wait before they are able to perform tasks that have been allocated to them. In the example in Table 8 the average waiting time will be the average of all the waiting times in Table 8 above the bottom line, which is equal to 9.8 minutes.
- Number of collisions avoided: Number of collisions avoided is exactly what the name of the objective suggests. It is the number of times the robots have to avoid a collision with another robot or an obstacle in their path.
- Maximum waiting time: Maximum total waiting time is the maximum between each robot's total waiting time. In the example showing the total waiting times in Table 8, robot two has the maximum total waiting time equal to 92 minutes.
- Average total waiting time: Average total waiting time is the average of each robot's total waiting time. In Table 8 it is the average of all the values in the bottom row ($Average(43, 92, 12)$) which is 49 minutes. Note here the difference between average waiting time and average total waiting time.

7.2 Data analysis evaluation setup and results

Kleyn (2020) considered five similar scenarios. Each scenario had 100 000 possible solutions. For each scenario, a data matrix had been generated for 8, 16, 24, 32, and 40 parts. The data set consisted of 100 000 rows and ten columns, one column per identified objective function. Each row represented a working shift (possible solution) and the columns represented each objective function listed previously. The data sets were investigated to analyse the correlation between the different objective functions. From the histogram in Figure 31, it can be seen for the largest problem of 40 parts that the increase in parts had a significant impact on the makespan. The rest of the average objective function values for the smaller problems are presented in Appendix A (Figures 65, 66, 67 and 68).

The rest of the objective functions remained constant with the increase in number of parts. The makespan, waiting time, and number of collisions increased as expected due to the increase in tasks. It is interesting to note that idle time stayed constant rather than increasing according to the number of parts.

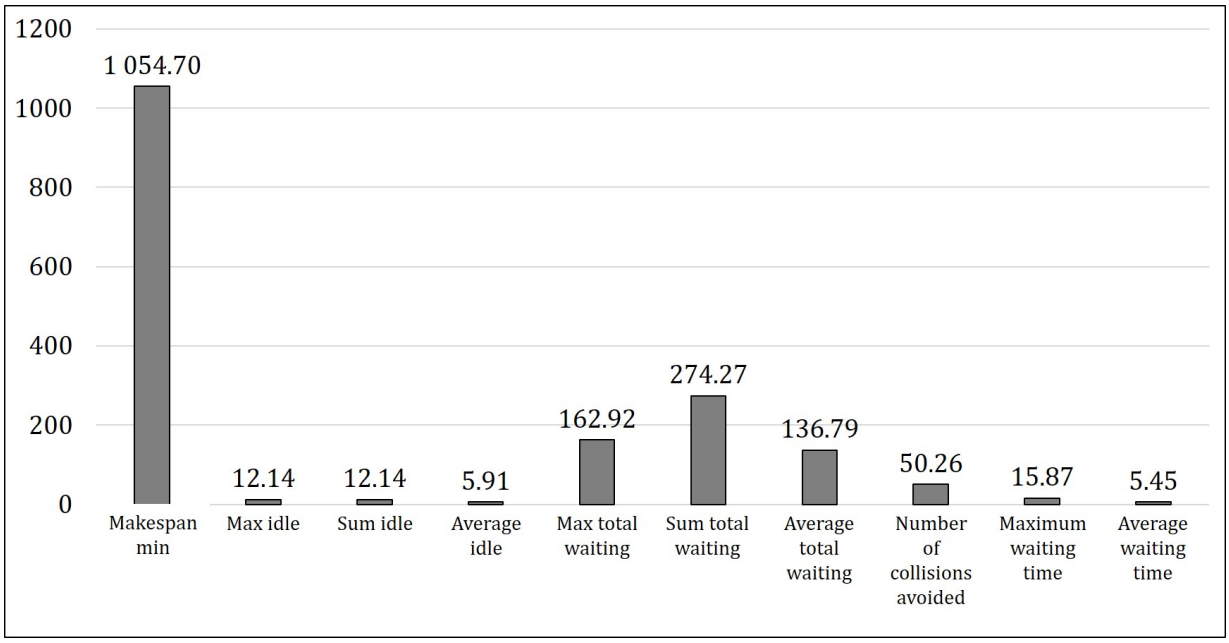

Figure 31: Average objective function values for 40 parts

7.3 Correlation coefficient results and interpretations

In order to understand the correlations between objective functions, scatter plot matrices (SPLOM) were constructed and analysed. From the SPLOM the correlation coefficients of all objectives with respect to each other were calculated. The correlations between different objective functions can be seen from the SPLOM. It also provides an indication of whether the correlation was strong or weak. The correlation plot matrices for the 40 part problem are shown in Figure 32. The correlation plot matrices for the smaller problems are presented in Appendix B (Figures 69, 70, 71 and 72).

SPLOMs and correlations were calculated for each problem size; however, they produced similar results, thus only one problem size is shown in Figure 32. It is notable that there are six instances of strong positive correlation. A strong correlation is evident when the correlation coefficient has a value greater than 0.9. The objective functions that showed strong correlation coefficients were:

- Makespan with respect to the sum of the total waiting time;
- Makespan with respect to the average total waiting time;
- Maximum idle time with respect to the sum of the total idle time;
- Maximum idle time with respect to the average idle time;
- Sum of the total idle time with respect to the average idle time;
- Sum of the total waiting time with respect to average total waiting time.

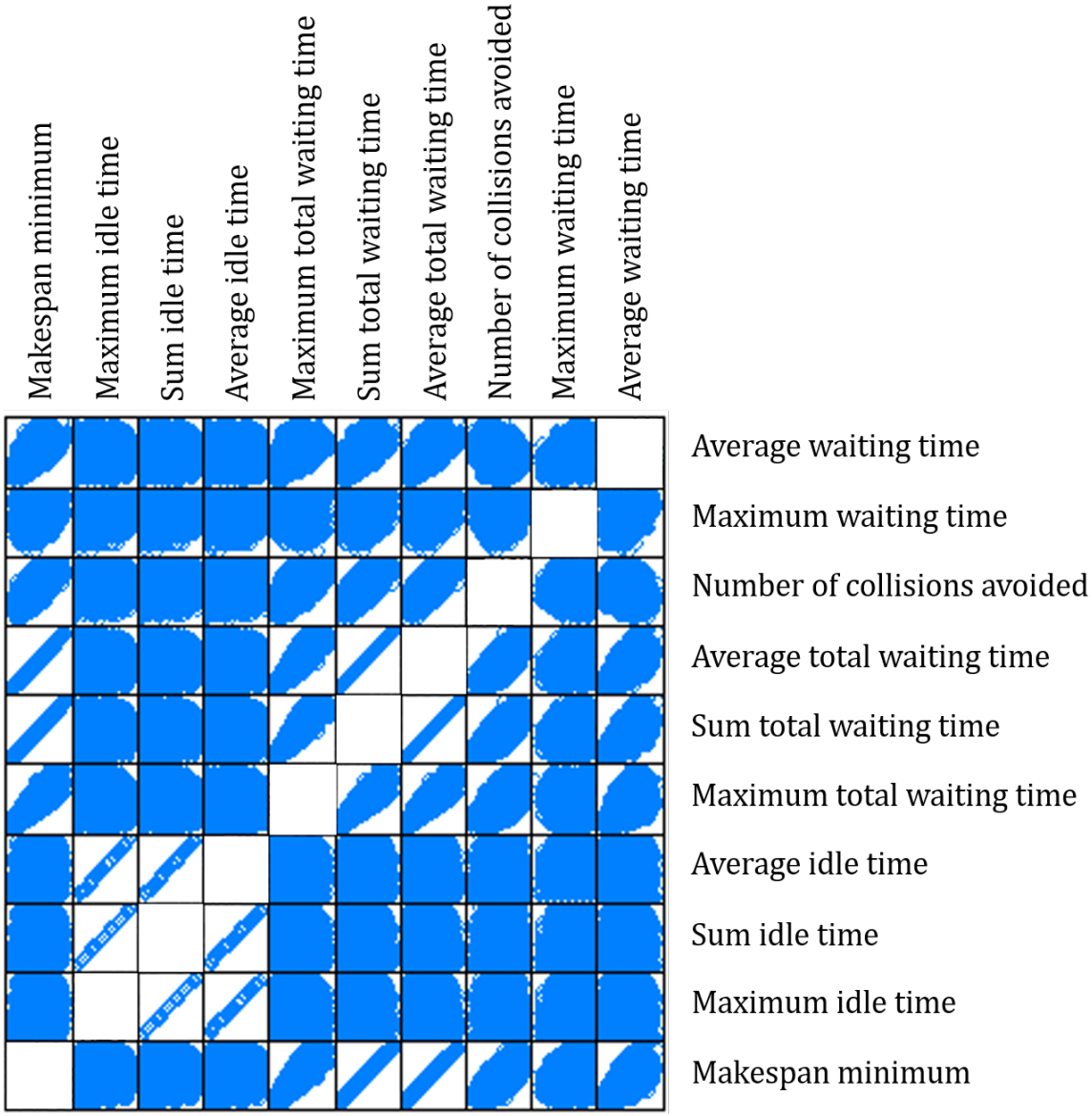

Figure 32: Correlation plot for 40 parts

Because the objective is to minimise the defined objective functions, the purpose of the correlation coefficient plots was to identify the correlations that are close to zero. The correlation coefficient tables for the four smaller problem sizes can be seen in Appendix C (Tables 15, 16, 17 and 18). The correlation coefficient table for the largest problem size can be seen in Table 9. The strong positive correlation objective functions indicate that there are no significant conflicts between the two objective functions. For this reason, these objective functions are not further investigated in this dissertation. Objective functions with correlation coefficients close to zero for all five data sets were investigated.

Table 9: Correlation coefficients for 40 parts

	Maximum idle time	Sum idle time	Average idle time	Maximum total waiting time	Sum total waiting time	Average total waiting time	Number of collisions avoided	Maximum waiting time	Average waiting time
Makespan minimum	0.15	0.15	0.15	0.73	0.98	0.98	0.68	0.21	0.63
Maximum idle time		1.00	1.00	-0.03	-0.03	-0.03	-0.02	0.03	-0.02
Sum idle time			1.00	-0.03	-0.03	-0.03	-0.02	0.03	-0.02
Average idle time				-0.03	-0.03	-0.03	-0.02	0.03	-0.02
Maximum total waiting time					0.74	0.74	0.51	0.03	0.47
Sum total waiting time						1.00	0.69	0.21	0.64
Average total waiting time							0.69	0.21	0.64
Number of collisions avoided								0.09	-0.12
Maximum waiting time									0.19

7.4 Objective space analysis for makespan versus number of collisions avoided

In this section the possible solution space was constructed using $100,000$ random sequences for all five data sets, and are presented in Figures 33, 34, 35, 36, and 37. The plotted fitness function values indicate the area that can be explored by the EMO algorithms for possible Pareto fronts of interest. The plotted fitness function values were analysed to determine the Pareto fronts. The Pareto fronts for each data set are highlighted in red.

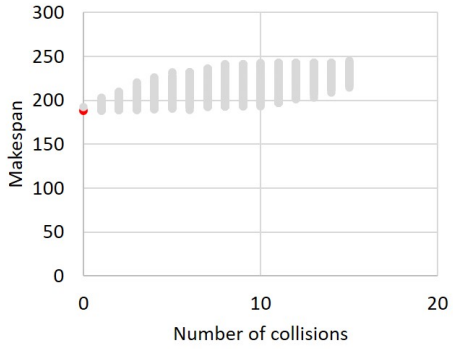

Figure 33: Makespan versus number of collisions objective space analysis for 8 parts

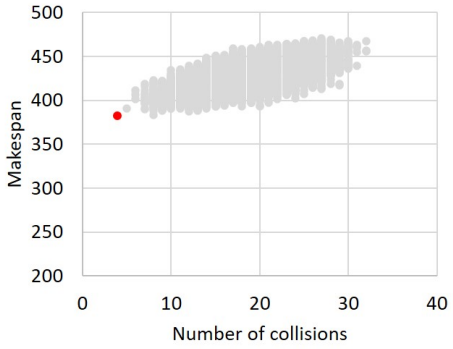

Figure 34: Makespan versus number of collisions objective space analysis for 16 parts

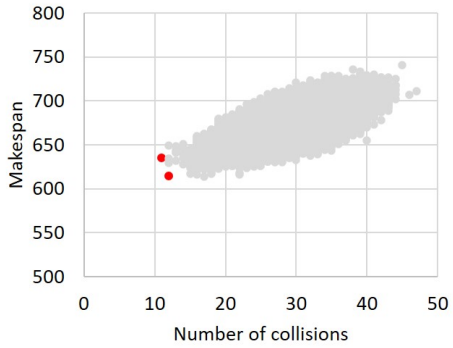

Figure 35: Makespan versus number of collisions objective space analysis for 24 parts

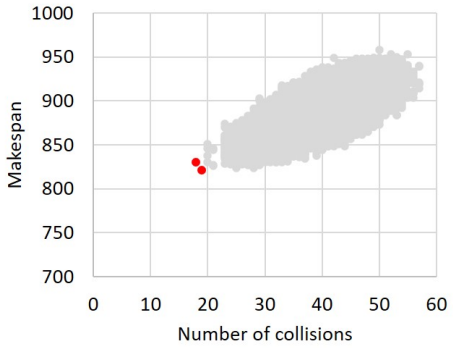

Figure 36: Makespan versus number of collisions objective space analysis for 32 parts

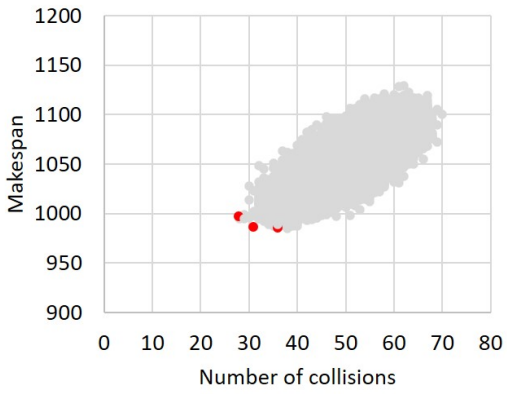

Figure 37: Makespan versus number of collisions objective space analysis for 40 parts

7.5 Summary

From the correlation coefficient analysis described in this chapter it was concluded that the two objective functions to consider for the multi-objective problem will be minimising makespan and minimising the sum of idle times. The next chapters focus only on the multi-objective problem, considering these two objectives.

Chapter 8

The multi-objective mathematical model and algorithm framework

In order to solve the multi-objective part sequencing and allocation problem a multi-objective mathematical model was developed. The following sections contain a comprehensive description of the problem. Section 8.1 presents the detailed mathematical model. Section 8.1 also discusses the function of each constraint. Section 8.2 describes the algorithm framework and its workings. Section 8.3 concludes the chapter.

8.1 Multi-objective part sequencing and allocation model

In a warehouse with I number of parts, all the parts need to be picked and binned on a daily basis. The picking and binning are done using a robot selected from a number of robots, K. A robot cannot carry more than one part at a time. A robot enters the warehouse through the entrance gate, completes the job (picking or binning), and then exits through the exit gate. The time it takes a robot to pick or bin a part is equal to the standard moving time plus the delay time for collision avoidance, the physical picking or binning time, and the time to travel to the exit gate.

The objectives of the model are to minimise the total time it takes all the robots to complete their picking and binning processes and to minimise the sum of idle times of all the robots. The model is driven by optimising the sequence and allocation of parts to be picked and binned. The sequence has a significant influence on the total picking and binning time. Each sequence has different planned path and collision avoidance scenario that must be incorporated. Changing the sequence affects the number of collisions to avoid and also influences the total picking and binning time.

This chapter was submitted to IEEE *Access*, Under review.

The following symbols need to be denoted:

$$y_{ijk} = \begin{cases} 1 \text{ if robot } k \text{ is binning or picking part } i \text{ immediately before part} \\ \quad j \text{ where } i,j \in \{1,...,I\} \text{ are the parts to be picked and} \\ \quad i,j \in \{I+1,...,2I\} \text{ are the parts to be binned,} \\ 0 \text{ otherwise} \end{cases}$$

$w_{ijk} \triangleq$ Time added to avoid collision if part i is picked or binned immediately before part j by robot k

$t_i \triangleq$ The transport time without delays of part i from entrance to exit

$n_i \triangleq$ Starting time of new task for part i from the entrance gate

$m_{ik} \triangleq$ The ending time of the last order picked, at point i for robot k

$p \triangleq$ Process time for picking or binning incurred for all parts

The model can be formulated as follows:

$$\text{Minimise } Z_1 = \max_{1\leq i\leq 2I}\{n_i\} \tag{61}$$

$$\text{Minimise } Z_2 = \sum_{k=1}^{\boldsymbol{K}}(\max_{1\leq i\leq 2I}\{n_i\} - \max_{1\leq i,j\leq 2I}\{m_{ik}\times y_{ijk}\}) \tag{62}$$

Subject to:

$$\sum_{i=1}^{2I}\sum_{k=1}^{K} y_{ijk} = 1 \quad \forall j \in \{1,...,2I\} \tag{63}$$

$$\sum_{j=1}^{2I}\sum_{k=1}^{K} y_{ijk} = 1 \quad \forall i \in \{1,...,2I\} \tag{64}$$

$$\sum_{i=1}^{2I} y_{ipk} - \sum_{j=1}^{2I} y_{pjk} = 0 \quad \forall p \in \{1,...,2I\}, k \in \{1,...,K\} \tag{65}$$

$$\sum_{k=1}^{K}\sum_{i=1,i\neq j}^{2I} y_{ijk}(n_i + p + w_{ijk} + t_i) \leq n_j \quad \forall j \in \{1,...,2I\} \tag{66}$$

$$\begin{aligned} t_i, n_i, m_{ik}, p, w_{ijk} &\in \mathbb{R} \\ y_{ijk} &\in \{0,1\} \end{aligned}$$

The first objective of the model is to minimise the maximum time it takes to complete all the picking and binning jobs for all the robots. The minimisation of objective one is done by taking the maximum ending times for all the robots of their last task iteration. The objective is to minimise the maximum ending times for the robots. The second objective of the model is to minimise the collective time that robots are idle. Equations (63) and (64) ensure that all parts are picked and binned, respectively. Each robot can only pick and bin one part at a time. This constraint is modelled using the same Equations (63) and (64). Equations (63) and (64) also assure that no two robots pick or bin the same part at the same time.

Equation (65) enforces continuity in the model, so that the next part is picked or binned. The different delay times that can be incurred, depending on the number of robots on the same route, is calculated using Equation (66). If there are robots on the route, Equation (66) will add the respective delay time given the position of the other robot. This strategy is used to avoid collisions in the warehouse.

8.2 Multi-objective algorithm

The algorithm framework presented for solving the single objective and multi-objective part sequencing and allocation problem is similar in structure and process. The same problem representation and procedure are used as for the multi-objective part sequencing and allocation problem. The only differences are that different evolutionary multi-objective optimisation algorithms are used and two objective functions are optimised simultaneously.

For solving the multi-objective part sequencing and allocation problem, four evolutionary multi-objective algorithms were selected and discussed earlier in this dissertation; namely, the MO-CMA-ES, MOPSO, MOEAD, and NSGA-III. The algorithm framework used to solve the multi-objective part sequencing and allocation problem is shown in Algorithm 16:

$I \triangleq$ Number of parts in the warehouse
$\boldsymbol{x} =$ Set of continuous variables given by the EMO algorithm framework where $\boldsymbol{x} = 3 \times I$
$\boldsymbol{p} =$ Set of parts to be picked per robot
$\boldsymbol{b} =$ Set of parts to be binned per robot
$\boldsymbol{a} =$ Set of parts allocated to a robot

Algorithm 16: Multi-objective part allocation and sequencing algorithm

```
Initialise a warehouse with I number of storage spaces
Divide the n_x-dimensional candidate solution into 3 sets
for The first set → picking sequence do
    Arrange the first set in ascending order → part picking sequence (p)
end
for The second set → binning sequence do
    Arrange the second set in ascending order → part binning sequence (b)
end
for The third set → robot allocation do
    Given the number of robots (K) in the system complete the allocation
end
Assign each robot → next destination node
for All the parts in the warehouse do
    Route each robot to their destination using the routing heuristic
    if the next node is occupied then
        WAIT
    end
    else
        Move along the path
    end
end
if robot is done with picking and binning then
    state = DONE
end
else
    state = ACTIVE
end
if all robot states = DONE then
    return Fval_1 = Count time steps used
    return Fval_2 = Sum of idle times
end
```

The EMO algorithm framework is similar to the single objective algorithm framework, the only difference being that the EMO algorithms are used and that the algorithm framework returns both objective function values. The algorithm framework only returns both objective function values if all the robots have completed their tasks. The process flow of the algorithm can be seen in Figure 38.

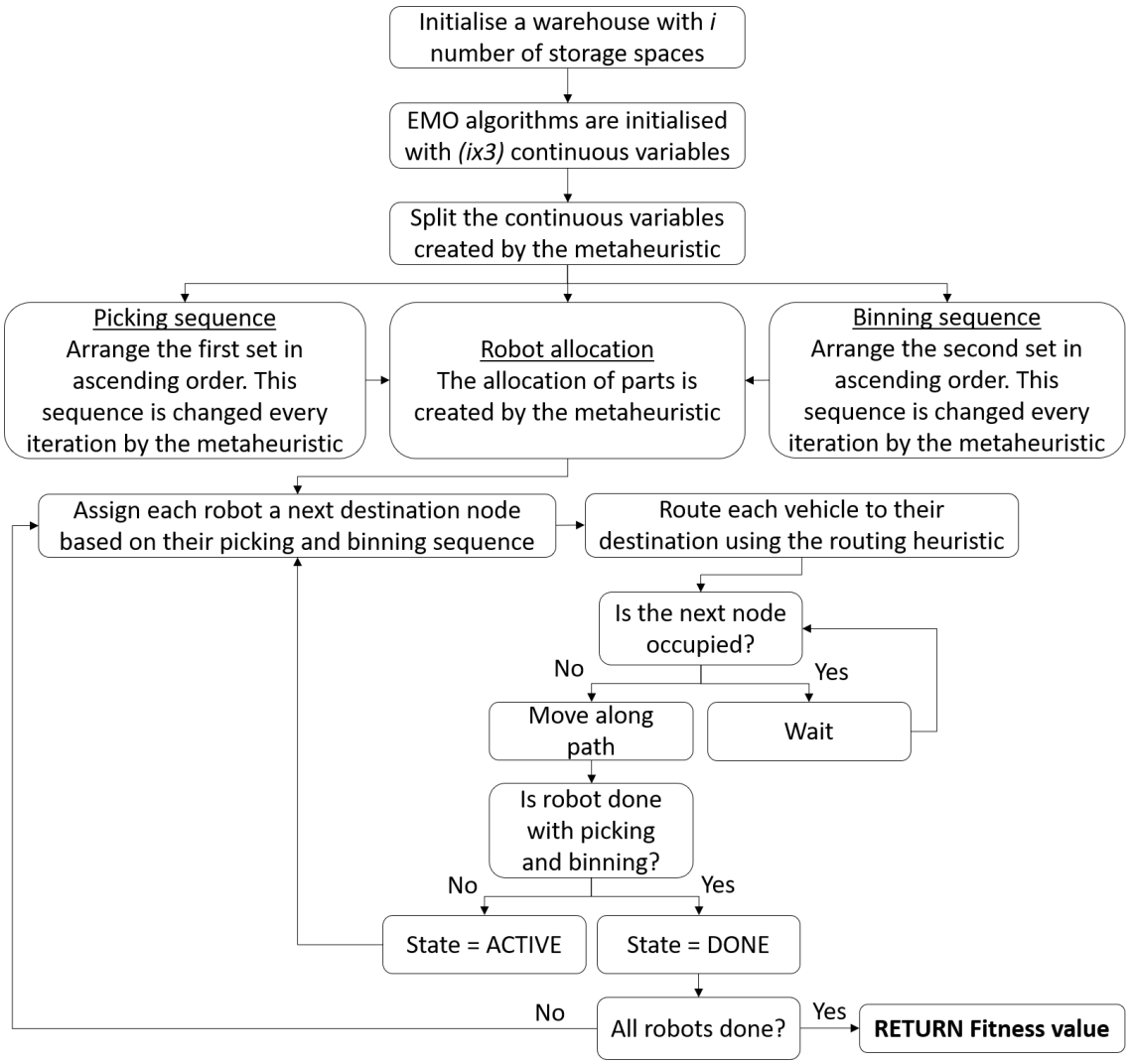

Figure 38: Process flow of multi-objective algorithm

8.3 Summary

This chapter discusses the multi-objective mathematical model in detail, with the relevant constraints. The pseudocode for the EMO algorithm framework for solving the part sequencing and allocation problem is also presented in this chapter. The next chapter discusses the empirical evaluation of the multi-objective problem in detail as well as the results of the multi-objective part sequencing and allocation problem.

Chapter 9

Empirical evaluation of the EMO algorithm framework

Given the multi-objective algorithm and EMO algorithms the empirical evaluation can be developed. In order to compare the performance of the EMO algorithms on the multi-objective part sequencing and allocation problem, performance metrics have to be determined for this EMO algorithm framework. Four popular EMO algorithms were tested in the multi-objective algorithm framework to determine whether feasible and promising solutions could be obtained by the algorithm framework. These algorithms were the NSGA-III, MOEAD, MOPSO, and MO-CMA-ES. The following sections discuss the EMO algorithm framework in more detail. Section 9.1 introduces the model parameters and the experimental setup, Section 9.2 presents the metric results for the defined multi-objective metrics. Section 9.3 explores the fitness values over iterations, HV, SM and number of Pareto points. Section 9.4 investigates the different Pareto fronts obtained from all thirty runs. The hypothesis test results are analysed in Section 9.5. The chapter is summarised in Section 9.6.

9.1 Experimental setup

The algorithm control parameters used in the empirical evaluation are provided in Table 10. The notation $x \rightarrow y$ is used to indicate that the associated parameter is decreasing or increasing linearly from x to y over 95% of the maximum number of iterations.

The EMO algorithm code used in the empirical evaluation includes the MOEAD with the Tchebycheff decomposition approach as found in Heris (2015a). The code for the MOPSO and NSGA-III are described in Heris (2015b) and Heris (2016) respectively. The MO-CMA-ES code that was used in this experimental setup is described in Hadka (2015).

The framework is able to accommodate any two fitness functions, however for this experimental setup the two objective functions considered are the minimising of makespan and minimising the sum of idle times.

This chapter was submitted to IEEE *Access*, Under review.

Table 10: Multi-objective model parameters

Parameter	Value used
General parameters	
Population size	100
Max iterations	1000
Number of independent simulation runs per problem size	30
MOPSO parameters	
Acceleration constant (c_1)	$2.0 \rightarrow 0.7$
Acceleration constant (c_2)	$0.7 \rightarrow 2.0$
Inertia weight	$0.9 \rightarrow 0.4$
MOEAD parameters	
Probability of reproduction	$1.0 \rightarrow 0.0$
NSGA-III parameters	
Probability of crossover	$0.6 \rightarrow 0.4$
Probability of mutation	$0.3 \rightarrow 0.0$
MO-CMA-ES parameters	
Sigma	0.5
Epsilons	0.05

9.2 Multi-objective metric results

An empirical evaluation of the EMO framework was conducted using the parameters described in Section 9.1. The results presented in this section include the results for both fitness values. The results for all the EMO algorithms are shown per performance metric and problem size.

First the average values over the thirty simulation runs for the hypervolume metric are analysed. From the results in Table 11 it is clear that the NSGA-III performed the best on average for the two largest data sets with regard to the hypervolume metric results. The MOPSO performed the best out of the three smaller data sets based on the hypervolume metric results. The best performing results are emboldened.

Table 11: Hypervolume results for all data sets

EMO Algorithm		8	16	24	32	40
NSGA-III	Average	809541	617993	209645	**207009**	**2121741**
	Standard deviation	2127	3802	5832	6511	14737
MOEAD	Average	806307	605604	372206	165821	2009629
	Standard deviation	2229	5502	7516	10376	20681
MOPSO	Average	**811139**	**618691**	**394310**	190277	2065129
	Standard deviation	1664	2868	5742	7010	17930
MO-CMA-ES	Average	803365	612398	386590	188738	2077875
	Standard deviation	9957	5534	9055	10145	17472

The hypervolume metrics were also plotted over time for one run. The results of the best HV metric per iteration can be seen in Figures 39 to 43. From the figures it can be noted that the MOEAD converges the fastest of all the algorithms. NSGA-III converges slightly slower, but does sometimes converge too quickly and becomes stuck in a local optima, as can be seen for the 24 parts problem. The MO-CMA-ES is the slowest converging algorithm of the four. In general, the NSGA-III and MOPSO convergence profiles seem to be the most suitable with regard to the hypervolume metric.

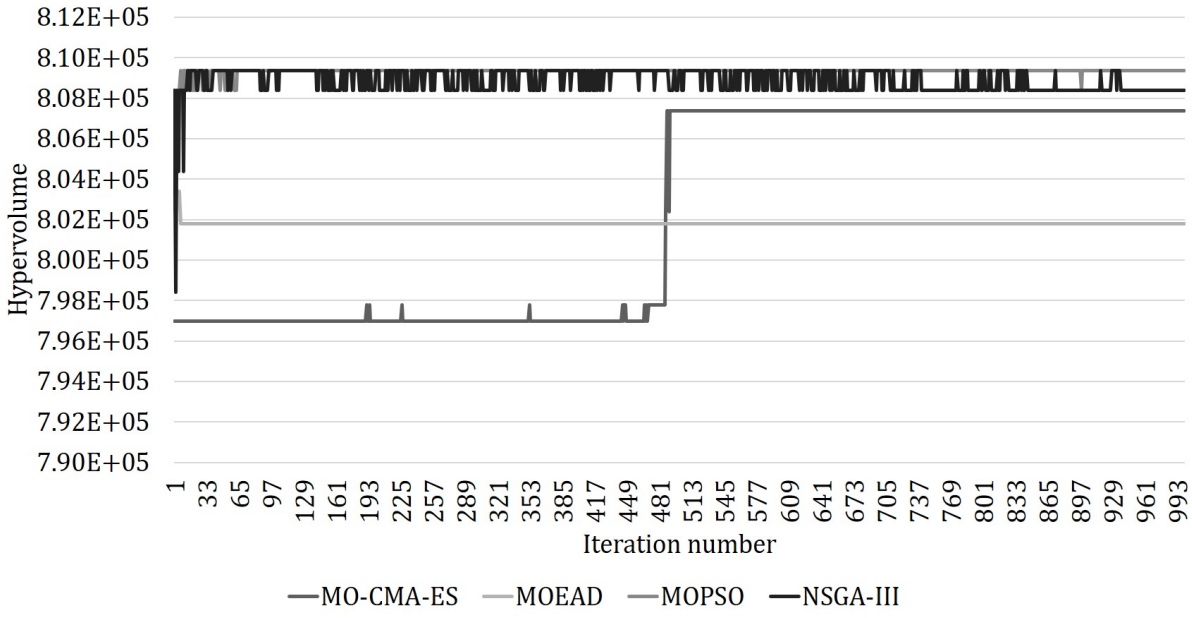

Figure 39: Hypervolume results for 8 parts over time, for one run

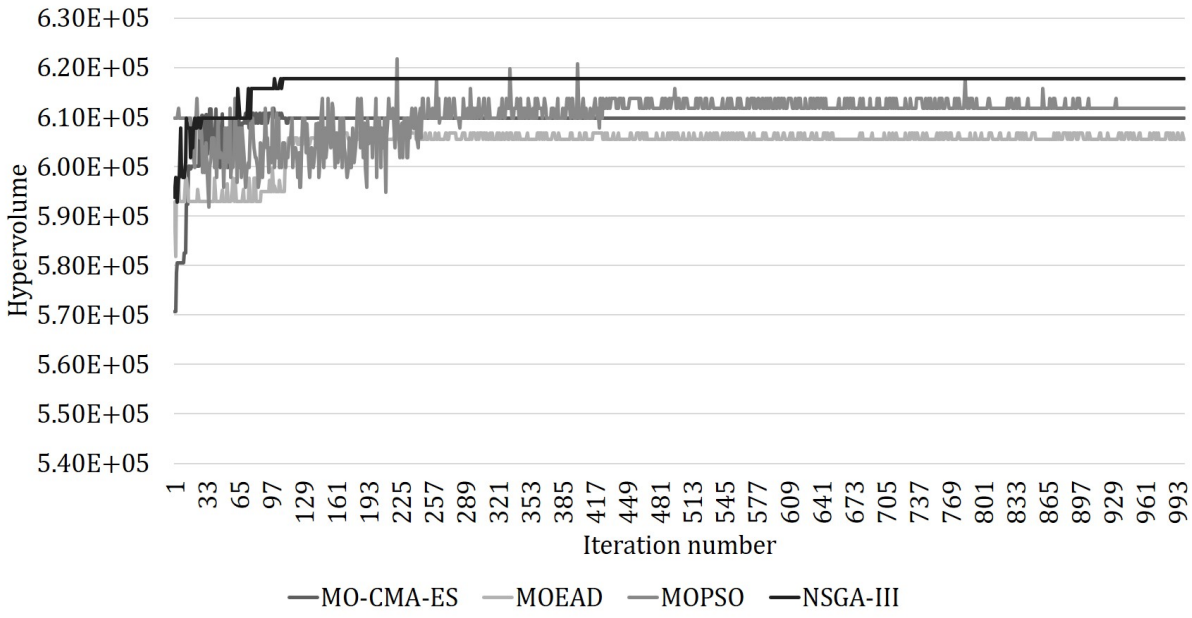

Figure 40: Hypervolume results for 16 parts over time, for one run

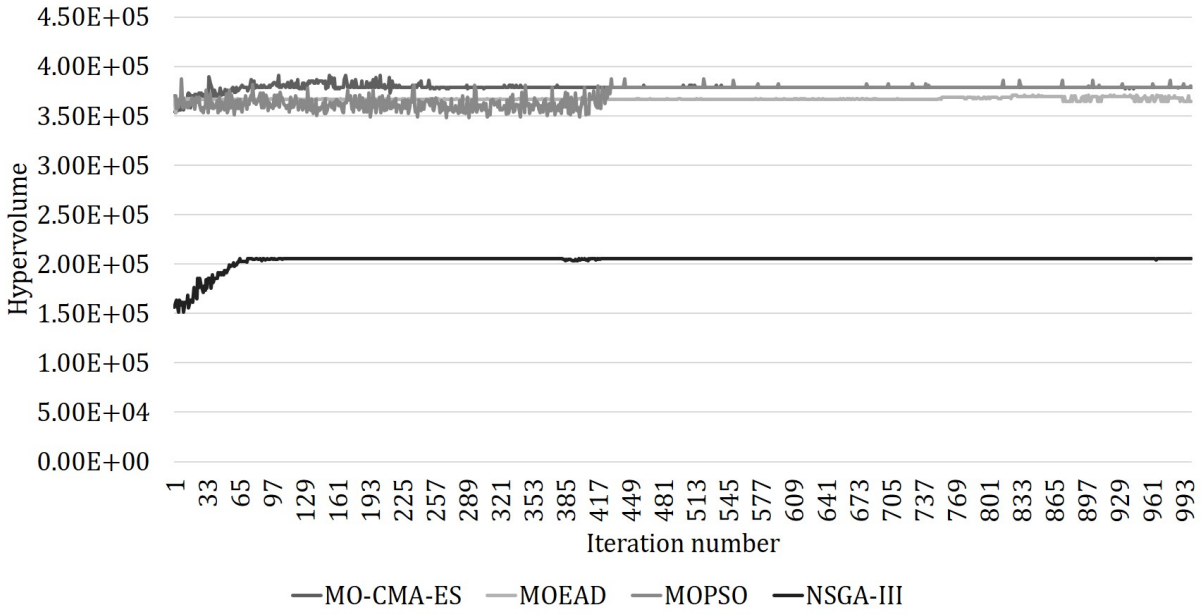

Figure 41: Hypervolume results for 24 parts over time, for one run

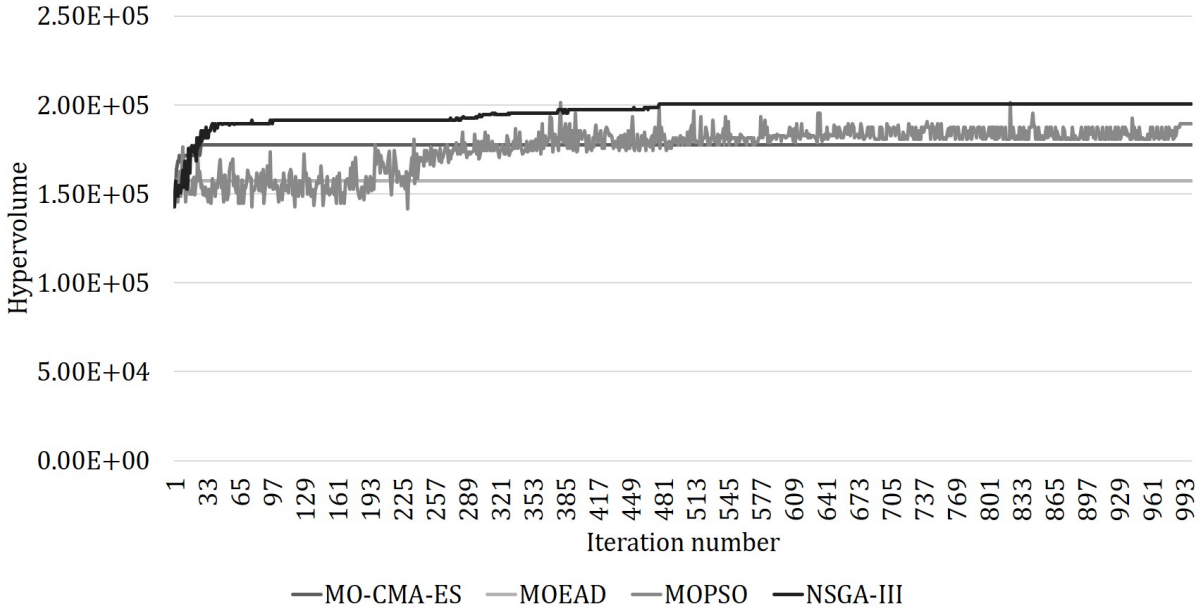

Figure 42: Hypervolume results for 32 parts over time, for one run

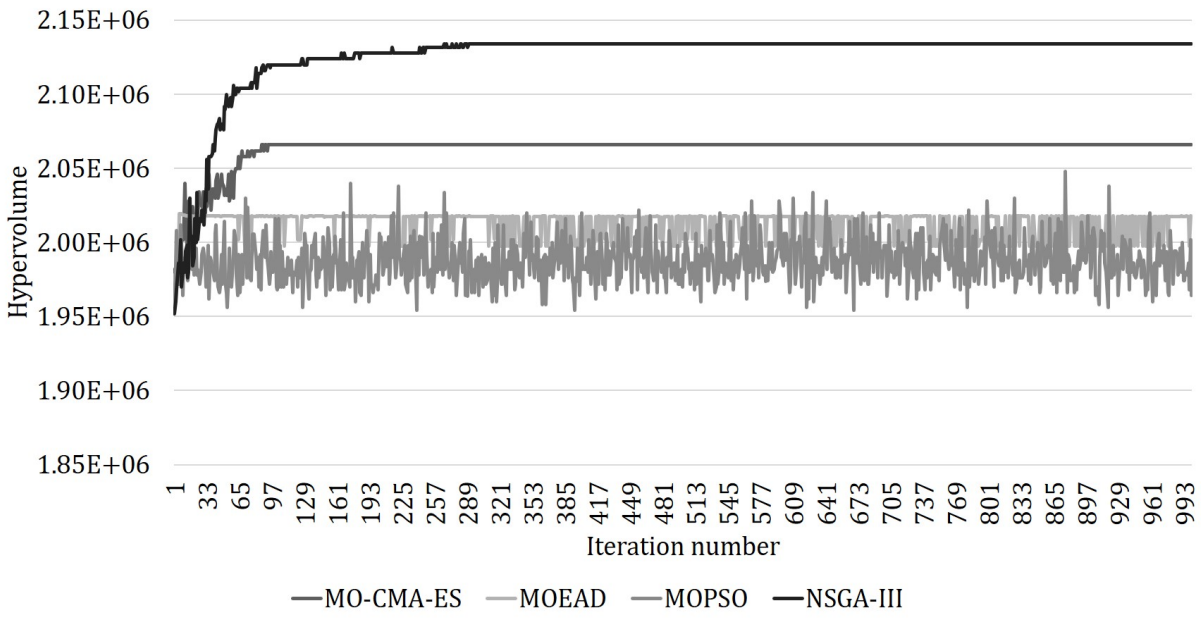

Figure 43: Hypervolume results for 40 parts over time, for one run

After the hypervolume metric results were analysed, the average values over thirty runs for the spread metric were analysed. From the results in Table 12 it is clear that the MOEAD performed the best on average for all the data sets with regards to the spread metric results.

Table 12: Spread results for all data sets

EMO Algorithm / Parts		8	16	24	32	40
NSGA-III	Average	0.10	0.05	0.27	0.39	0.00
	Standard deviation	0.38	0.30	1.06	2.12	0.00
MOEAD	Average	**0.21**	**1.09**	**3.09**	**1.89**	**4.63**
	Standard deviation	0.82	2.80	3.48	3.79	5.05
MOPSO	Average	0.05	0.05	0.00	0.13	0.09
	Standard deviation	0.20	0.28	0.00	0.53	0.46
MO-CMA-ES	Average	0.14	0.44	0.84	0.14	2.53
	Standard deviation	0.52	1.39	2.32	0.74	7.34

The SM plots are shown in Figures 44 to 48. The results were less informative. For the eight part problem there was almost no spread visible; for the sixteen part problem the spread increased slightly at the start of the run but also faded with no spread at the end of the run. For the twenty-four part problem there was spread visible at the start of the run, which continued at a low rate to the end with a slight increase for some EMO algorithms at the end of the run. The spread observed for the thirty-two part problem was only at the start of the run. The forty-part problem had the most spread throughout the entire run.

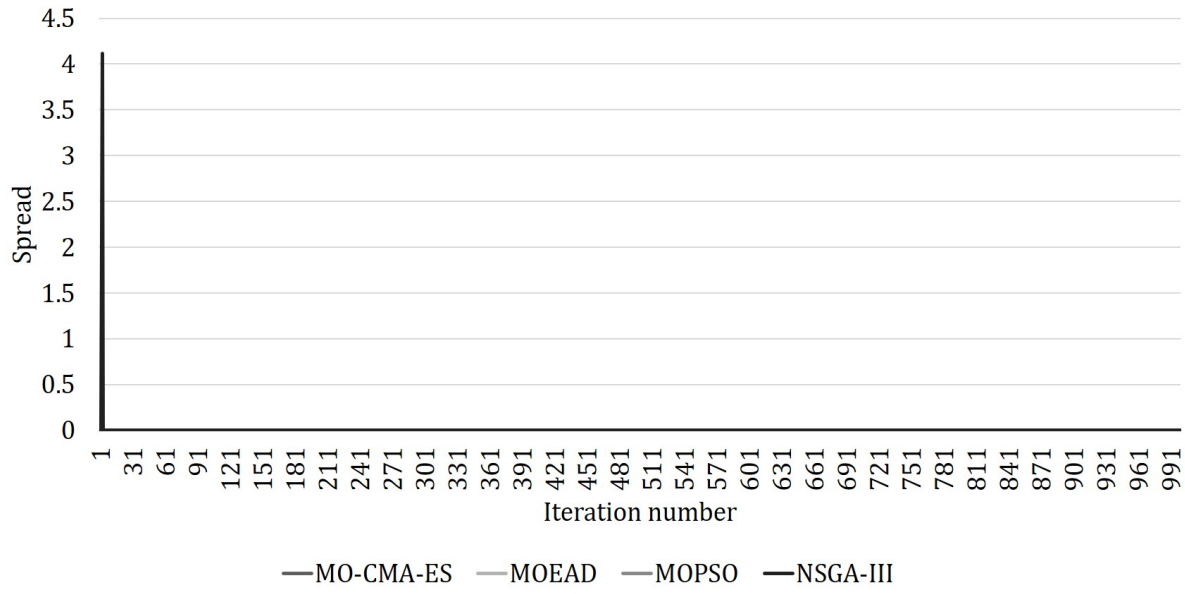

Figure 44: Spread results for 8 parts over time, for one run

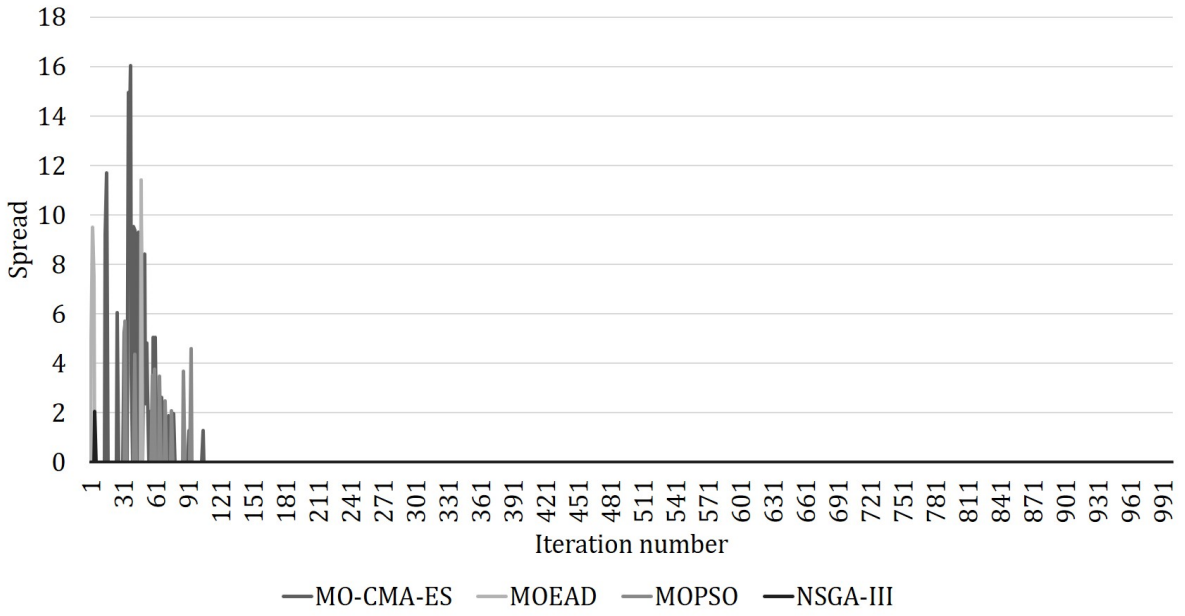

Figure 45: Spread results for 16 parts over time, for one run

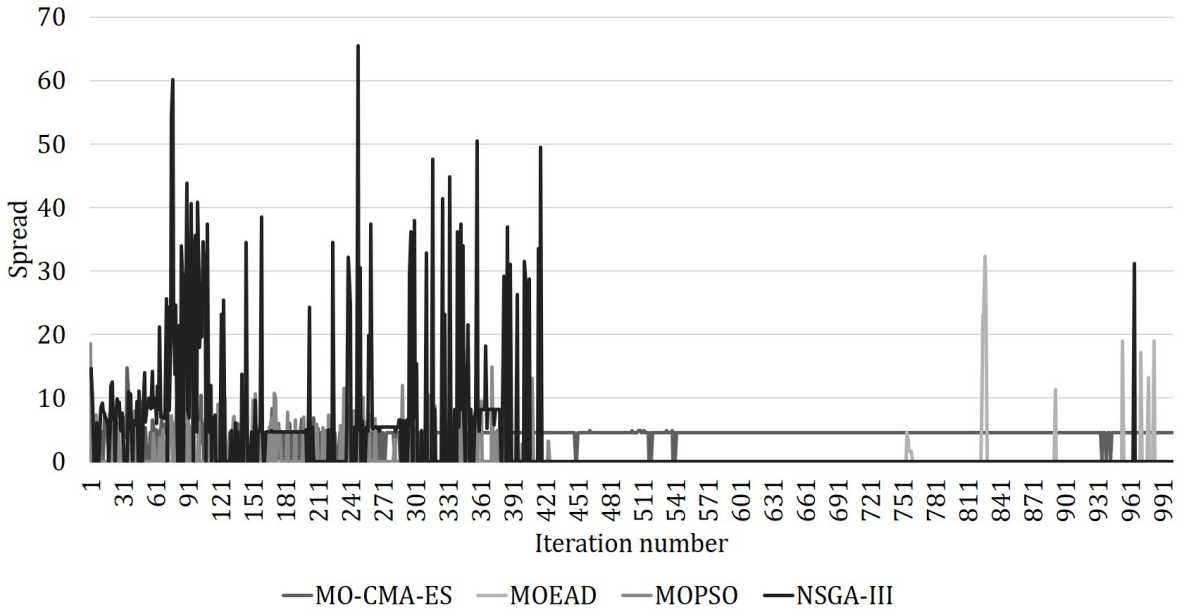

Figure 46: Spread results for 24 parts over time, for one run

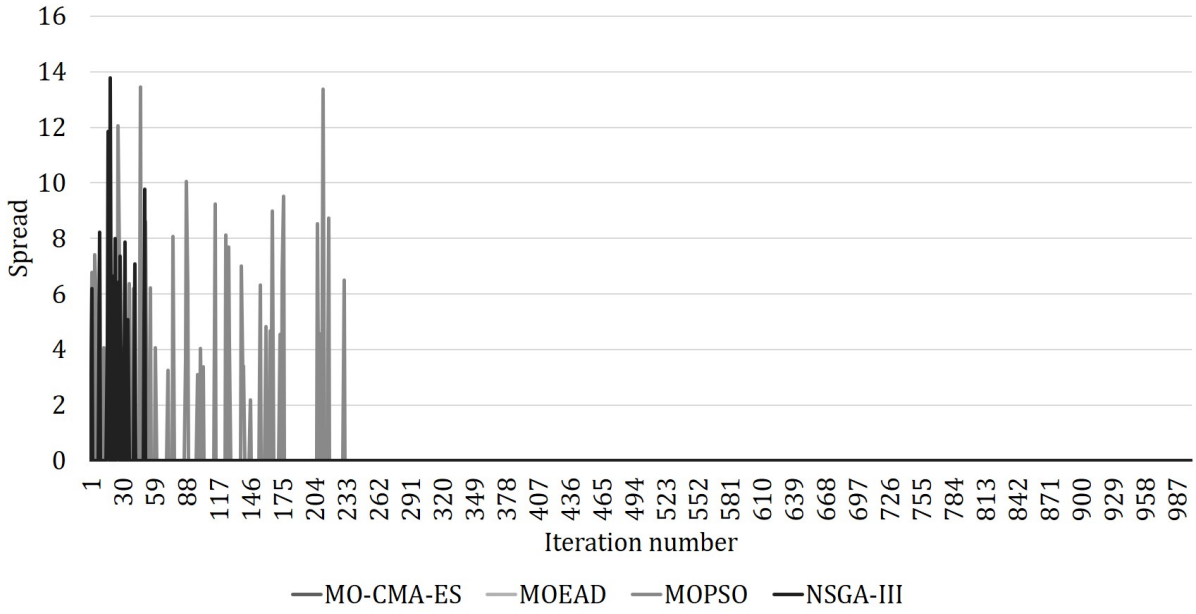

Figure 47: Spread results for 32 parts over time, for one run

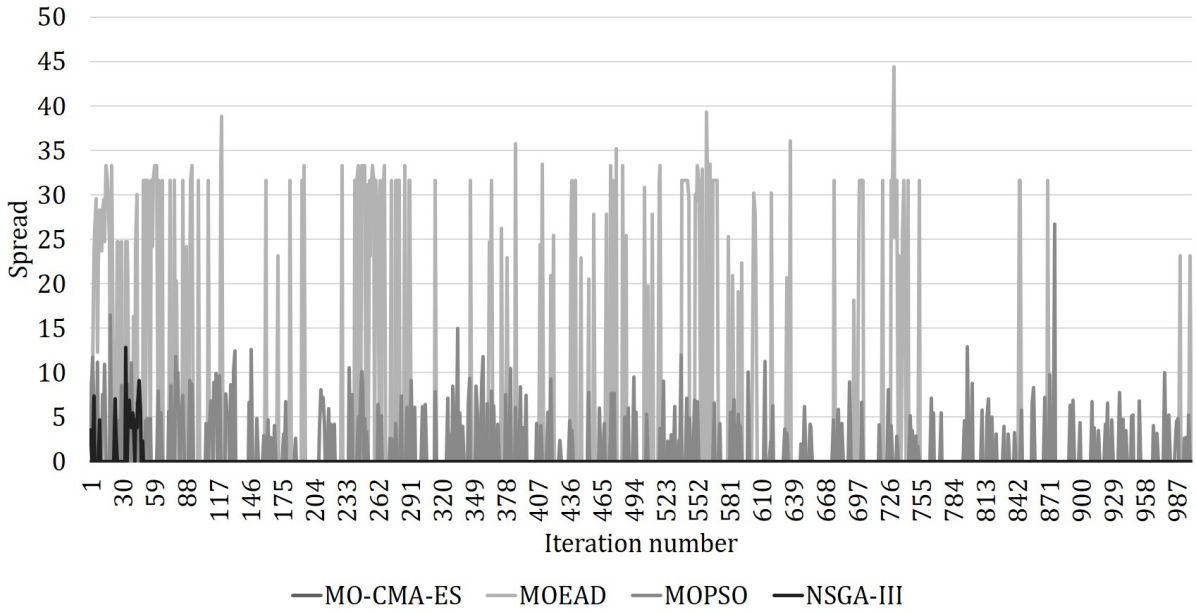

Figure 48: Spread results for 40 parts over time, for one run

The last results that were analysed were the average values over thirty runs for the number of Pareto solutions metric results. From the results in Table 13 it is clear that the MOEAD, on average, performed the best for all the data sets with regard to the number of Pareto solutions metric results.

Table 13: Number of Pareto solutions, results for all data sets

EMO Algorithm / Parts		8	16	24	32	40
NSGA-III	Average	1.50	1.27	1.23	1.37	1.23
	Standard deviation	0.62	0.51	0.56	0.80	0.42
MOEAD	Average	**1.73**	**1.47**	**2.40**	**1.83**	**2.67**
	Standard deviation	0.57	0.81	1.14	0.78	1.30
MOPSO	Average	1.60	1.23	1.67	1.37	1.33
	Standard deviation	0.61	0.50	0.38	0.60	0.54
MO-CMA-ES	Average	1.50	1.37	1.87	1.27	1.57
	Standard deviation	0.62	0.66	1.06	0.51	0.72

The number of Pareto solutions plots are shown in Figures 49 to 53. The results for the number of the Pareto points did not show any pattern or trend, apart from the fact that the number of Pareto solutions never exceeded six.

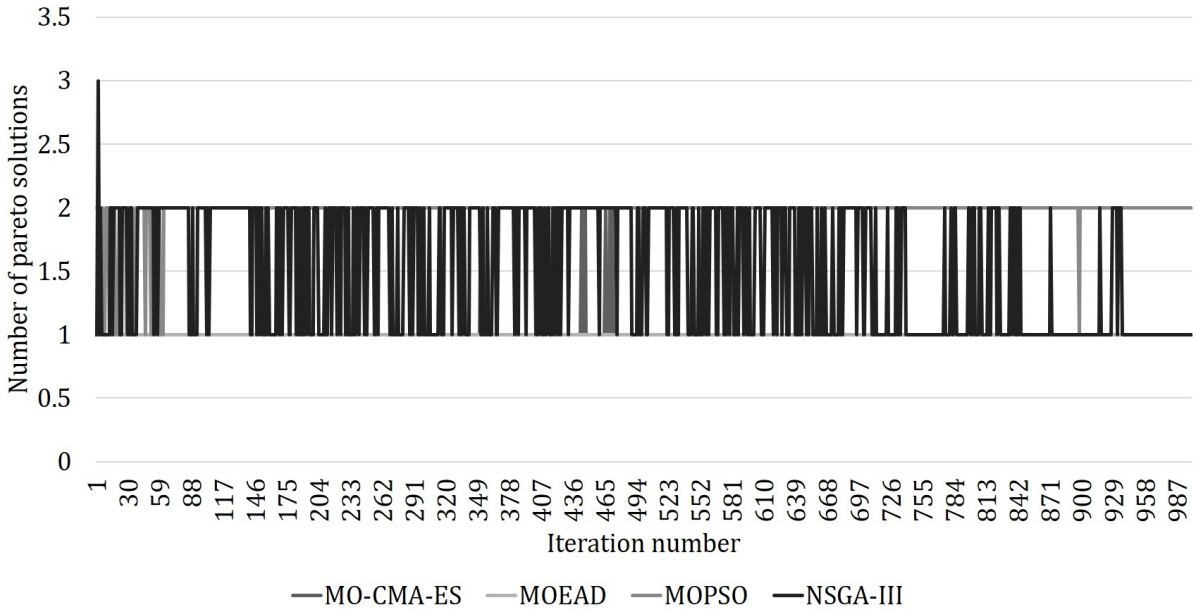

Figure 49: Number of Pareto solutions results for 8 parts over time, for one run

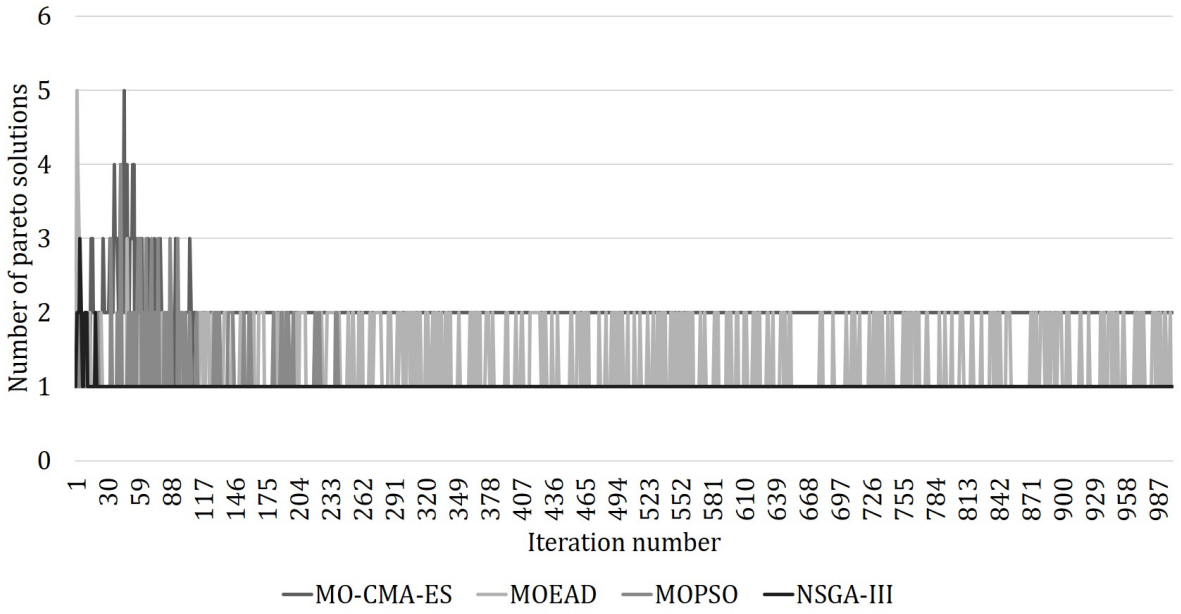

Figure 50: Number of pareto solutions results for 16 parts over time, for one run

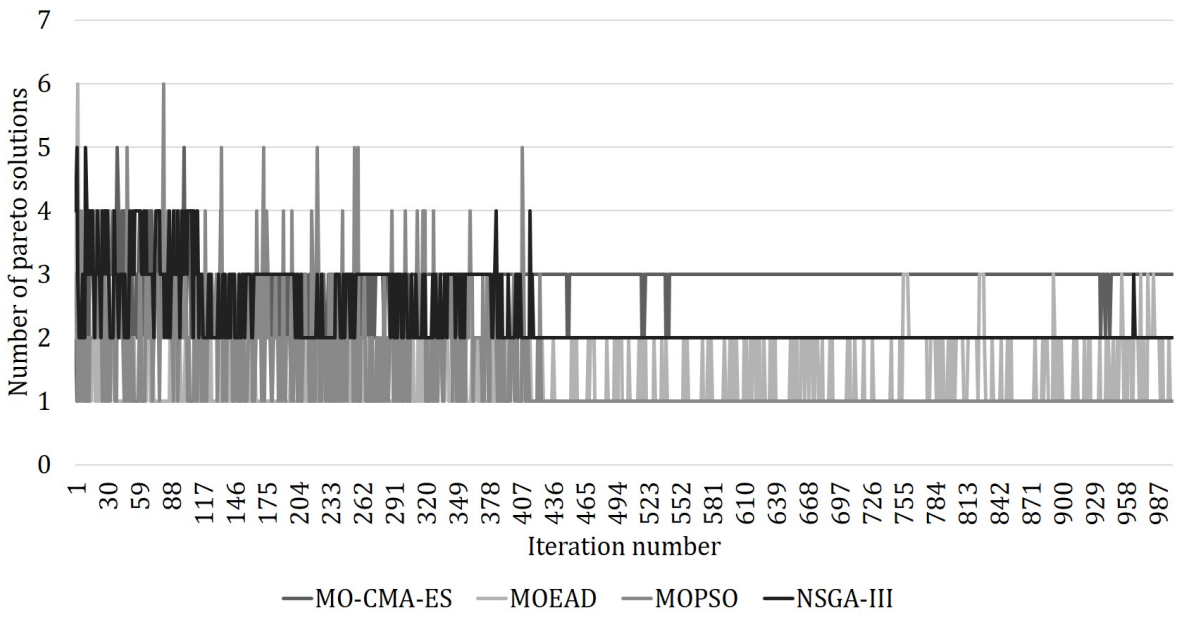

Figure 51: Number of pareto solutions results for 24 parts over time, for one run

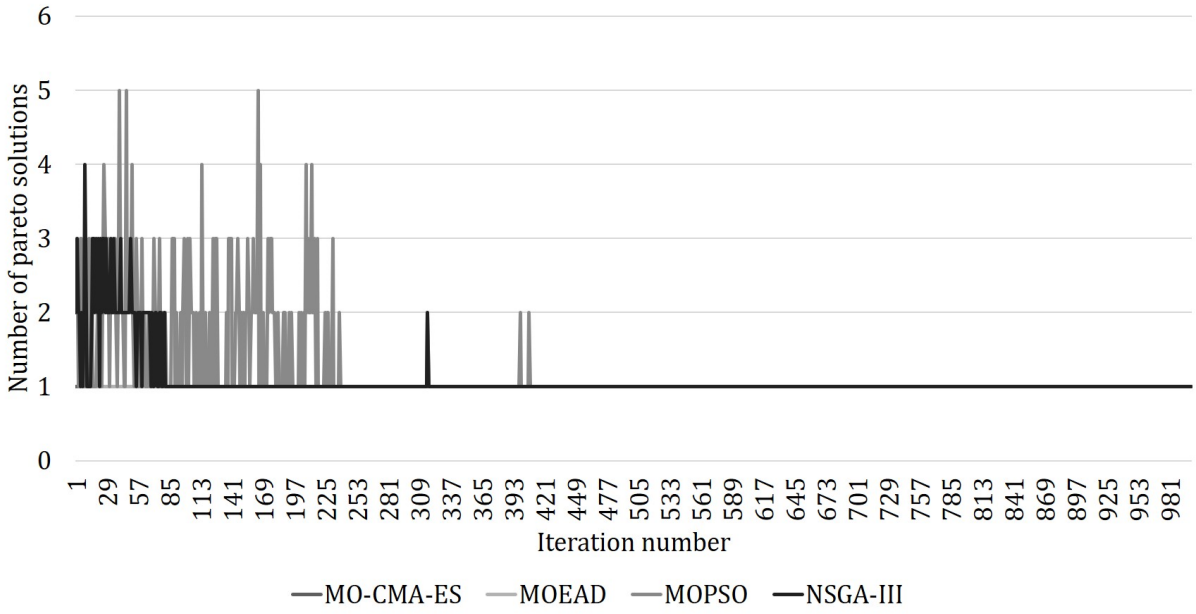

Figure 52: Number of pareto solutions results for 32 parts over time, for one run

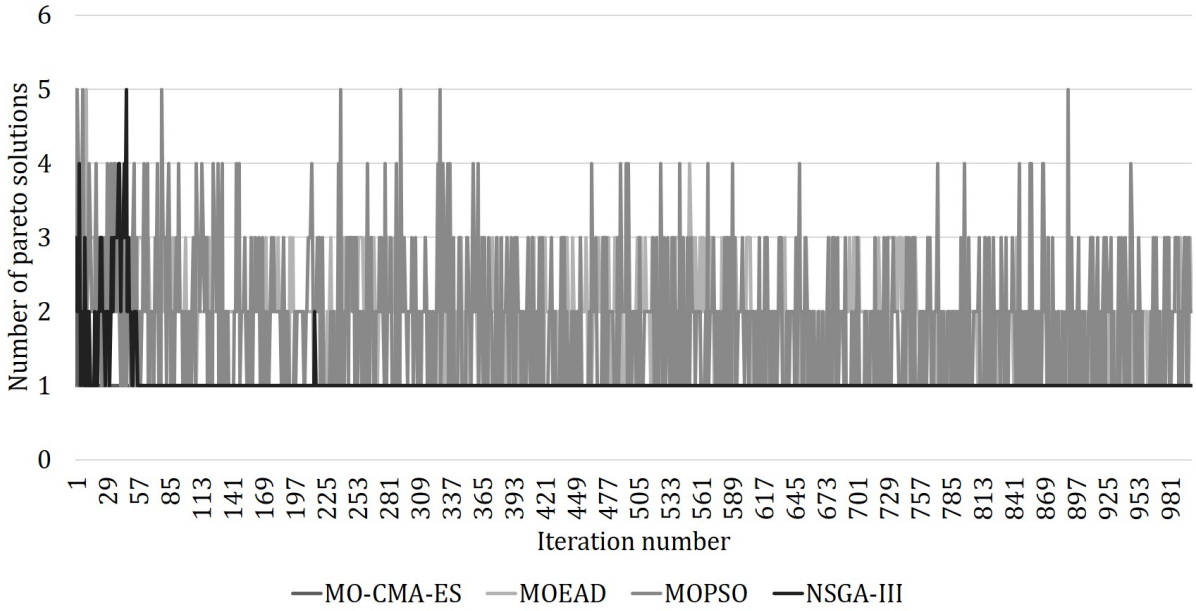

Figure 53: Number of pareto solutions results for 40 parts over time, for one run

The first important observation was that feasible and promising results could be obtained by the EMO algorithm framework. Furthermore, it could be concluded that for the HV metric there were two algorithms that performed well. For the smaller data sets the MOPSO algorithm performed the best and for the larger data sets the NSGA-III performed the best. The MOEAD performed the best on all data sets with regard to the SM and NPS metrics.

9.3 Multi-objective EMO algorithm performance analysis

The top five graphs in Figure 54 show the HV metric versus iterations for all four EMO algorithm considered. The HV metric was calculated using a reference point that is greater than both the objective values on the x-axis and y-axis. A higher HV is more desirable for the multi-objective part sequencing and allocation problem. Given the HV results for the 8 and 16 part problem it can be see that the MO-CMA-ES, NSGA-III and MOPSO performed relatively well compared to the MOEAD that struggled to achieve a good HV value. As the problem size increased the algorithms responded differently, in the specific run presented the NSGA-III struggled to find a good solution for the 24 part problem, but performed very well against the other EMO algorithms on the 32 and 40 part problem. The MOPSO and MO-CMA-ES kept on performing well as the problem size increased. The MOPSO performed well with respect to the HV metric which indicates that the erratic search observed on the single objective problem might not negatively affect the MOPSO in the multi-objective search space. Also, the number of duplicate solutions found by the single objective GCPSO did not have such a significant impact on the MOPSO. The MOEAD was not able to outperform all of the EMO algorithms on any of the problem sizes.

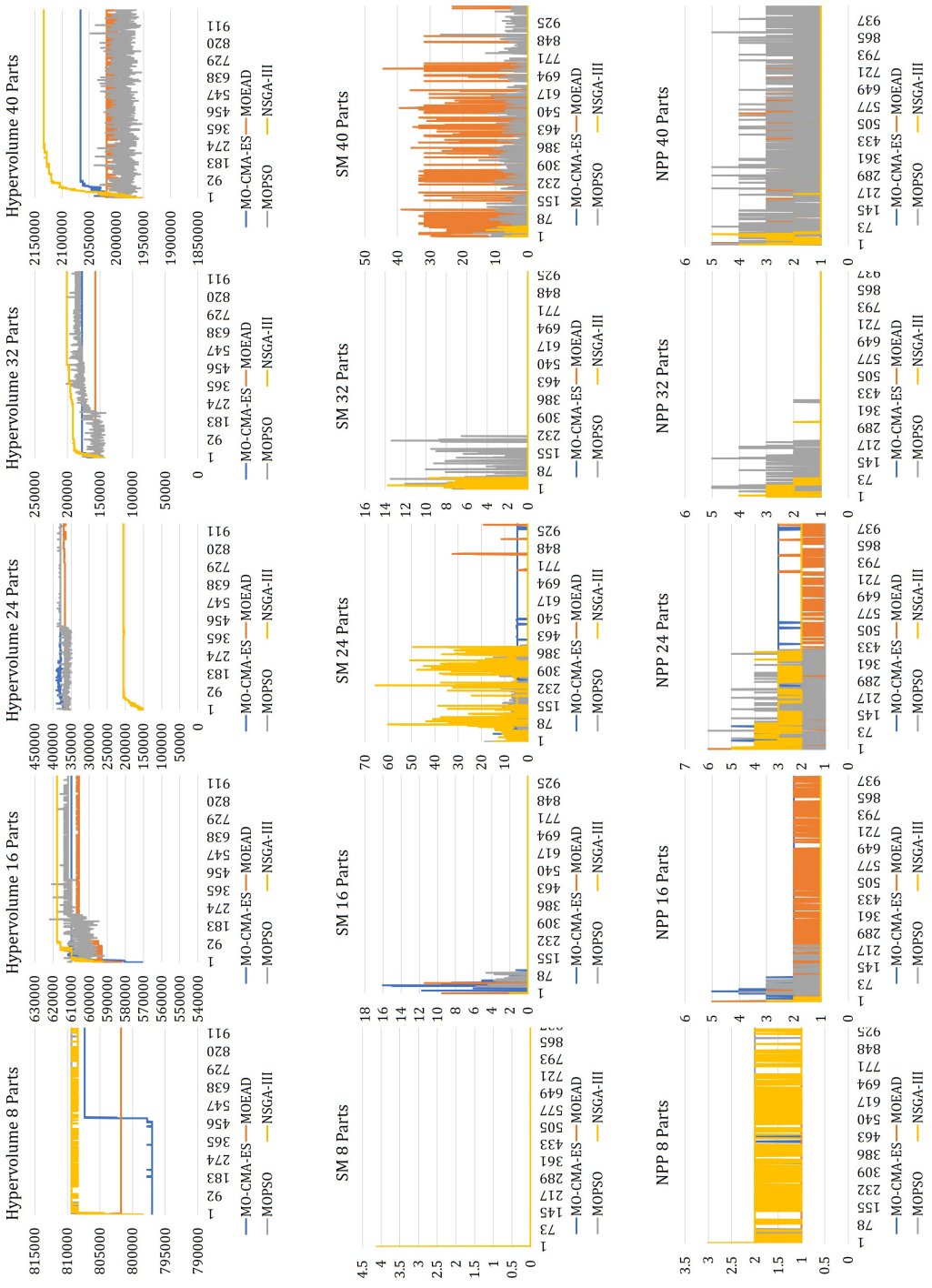

Figure 54: Multi-objective EMO algorithm results analysis

The spread metric for the five data sets are shown in the middle row of Figure 54. The spread of the solution gives an indication of the spacing between the pareto points. From the spread metric results it can be seen that there is a definite trend with regard to problem size versus solution spread over iterations. As the problem size increased the spread remained higher for longer during the run. Considering the largest problem, 40 parts, it is noticeable that the MOEAD has achieved a much greater spread over the iterations than any other EMO algorithm considered.

The number of pareto solutions versus iterations are depicted for all five data sets in the bottom five graphs in Figure 54. The number of pareto points metric is the exact number of pareto points in the pareto front. The number of pareto points are used to determine which EMO algorithm delivered the most pareto points.

The results obtained from the EMO algorithms are supported by the work presented by Gu et al. (2021). Gu et al. (2021) investigated the performance of a surrogate-assisted MOPSO on constrained combinatorial optimisation problems. The results showed that the MOEAD was not able to match the performance of the NSGA-II and MOPSO with regards to HV. The same results were observed in this dissertation where the MOEAD was the worst performing multi-objective algorithm in terms of the HV metric. The same results were observed in Got et al. (2020), where the results also showed that the MOPSO outperformed the MOEAD on the HV metric.

The second observation from the results indicated that the MOEAD is good at finding pareto fronts with larger spreads and number of pareto points, indicating that the MOEAD was better at finding solutions at the ends of the pareto front. The mechanism that MOEAD uses to decompose the multiple-objective into single objectives clearly enables the algorithm to find a pareto front with high spread. This result was also confirmed by Azadeh et al. (2017), where it was observed that the MOEAD showed superiority for large size problems based on the number of pareto solutions, spacing, and diversity.

9.4 Multi-objective EMO algorithm Pareto front analysis

In this section one random Pareto front from the 30 runs have been plotted for all five data sets to illustrate the spread, position and number of Pareto solutions.

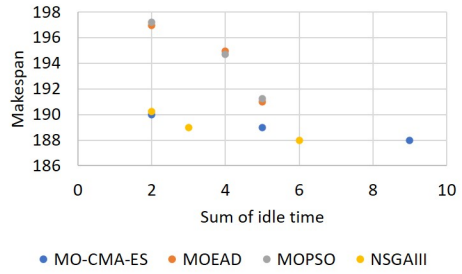

Figure 55: Pareto fronts for one run (8 Parts)

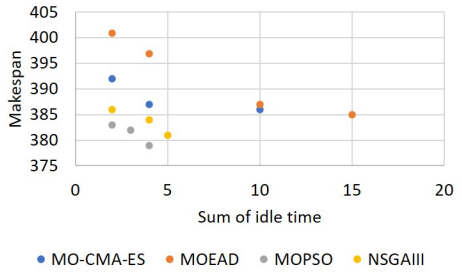

Figure 56: Pareto fronts for one run (16 Parts)

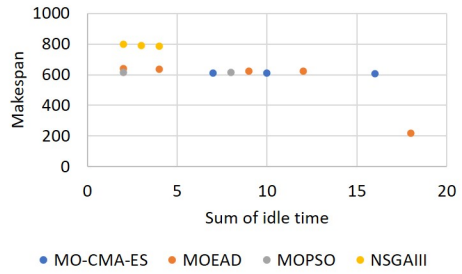

Figure 57: Pareto fronts for one run (24 Parts)

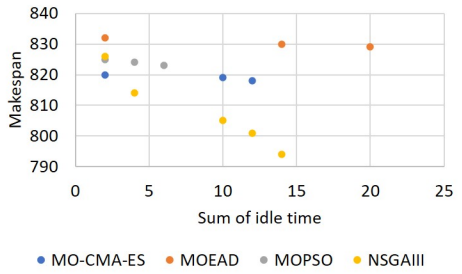

Figure 58: Pareto fronts for one run (32 Parts)

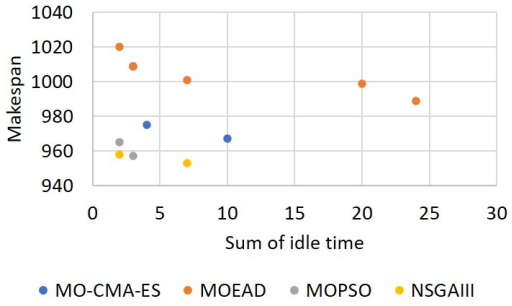

Figure 59: Pareto fronts for one run (40 Parts)

The Pareto solutions presented in Figure 55, for the 8 part problem indicate that between the NSGA-III and the MO-CMA-ES the best Pareto front were found. The Pareto points for the MOPSO and MOEAD in this instance were further from the point $(0, 0)$ an would therefor yield a smaller HV value which is not necessarily desirable. From Figure 56, the 16 part problem, the MOPSO algorithm performed the best with a Pareto front lower than all the other Pareto fronts plotted. The 24 part problem does not present an obvious best performing EMO algorithm. This is the reason why the metrics are calculated based on HV, SM, and NPP to determine based on each metric, which EMO algorithm performed the best. From the 32 and 40 part problem it is noticeable that the Pareto fronts from the different EMO algorithms start to perform differently, indicating that some algorithms are performing better than others. It is notable in the 40 problem that the MOEAD showed a greater spread with respect to the other EMO algorithms.

All the Pareto fronts and points for all 30 experimental runs are presented in Figures 60, 61, 62, 63, and 64. The main purpose of these figures are to identify characteristics from the EMO algorithms such as their Pareto point spread, the number of Pareto points and grouping over the 30 runs. From Figure 60 it is visible that the MO-CMA-ES and MOEAD had the greatest spread of Pareto points. For the 16 part problem the MOEAD, MO-CMA-ES and MOPSO showed more spread of Pareto points. The 24 problem had very similar Pareto points from the MOEAD, MO-CMA-ES and MOPSO, all with a decent spread. The NSGA-III was not able to find competitive solutions with respect to the MO-CMA-ES, MOPSO and MOEAD. Looking at the Pareto solution results for the 32 and 40 part problem, the points were definitely more spread out than with the smaller problems. One observation that can be made for these two problems are that the NSGA-III, MOPSO and the MO-CMA-ES had the greatest density of its Pareto points closest to the point $(0, 0)$.

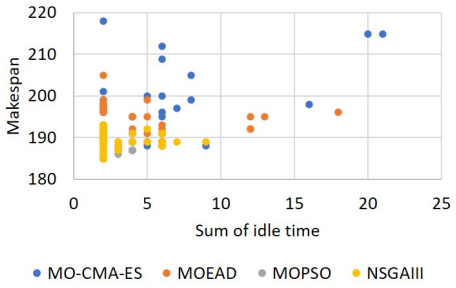

Figure 60: All Pareto points for 30 runs (8 Parts)

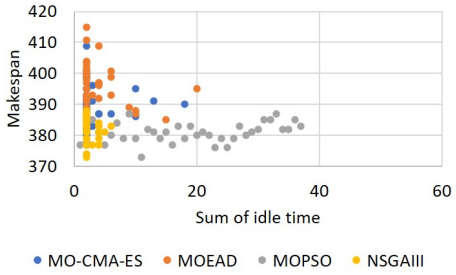

Figure 61: All Pareto points for 30 runs (16 Parts)

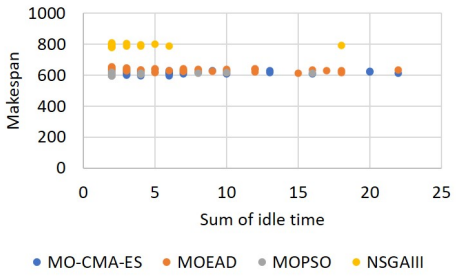

Figure 62: All Pareto points for 30 runs (24 Parts)

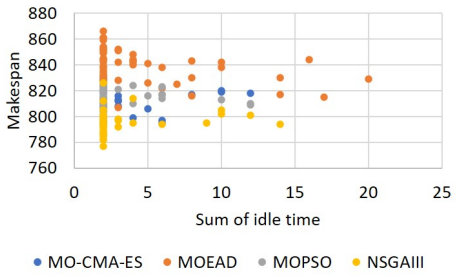

Figure 63: All Pareto points for 30 runs (32 Parts)

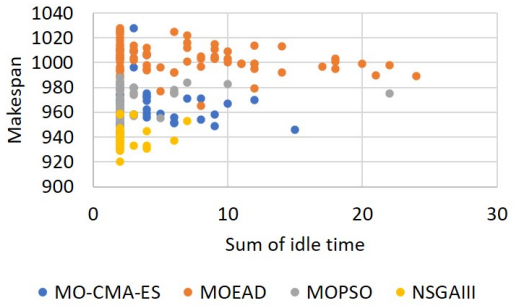

Figure 64: All Pareto points for 30 runs (40 Parts)

9.5 EMO algorithms hypothesis test

The results of a statistical comparison in Table 14 were obtained by comparing the result of an algorithm's performance to each of the other algorithms' performance based on the HV, SM, and NPS metrics, for each data set. For every comparison, a Mann–Whitney U test at 5% significance was performed (using the two sets of 30 sequences of the two algorithms under comparison) and if the first algorithm outperformed the second algorithm statistically significantly, a win was recorded for the first algorithm. If no statistical difference could be observed, a draw was recorded. If the second algorithm outperformed the first algorithm, a loss was recorded for the first algorithm. The total number of wins, draws, and losses was then recorded for all data sets of the algorithm under evaluation.

Table 14: Hypothesis test results for HV, SM and NPS for each EMO algorithm

		MO-CMA-ES	MOPSO	MOEAD	NSGA-III	Total
HV	**MO-CMA-ES**	-	1-1-3	4-1-0	1-0-4	6-2-7
	MOPSO	3-1-1	-	5-0-0	2-1-2	**10-2-3**
	MOEAD	0-1-4	0-0-5	-	1-0-4	1-1-13
	NSGA-III	4-0-1	2-1-2	4-0-1	-	10-1-4
		MO-CMA-ES	**MOPSO**	**MOEAD**	**NSGA-III**	**Total**
SM	**MO-CMA-ES**	-	1-4-0	0-2-3	1-4-0	2-10-3
	MOPSO	0-4-1	-	0-3-2	0-5-0	0-12-3
	MOEAD	3-2-0	2-3-0	-	3-2-0	**8-7-0**
	NSGA-III	0-4-1	0-5-0	0-2-3	-	0-11-4
		MO-CMA-ES	**MOPSO**	**MOEAD**	**NSGA-III**	**Total**
NPS	**MO-CMA-ES**	-	1-4-0	0-2-3	1-4-0	2-10-3
	MOPSO	0-4-1	-	0-2-3	0-5-0	0-11-4
	MOEAD	3-2-0	3-2-0	-	3-2-0	**9-6-0**
	NSGA-III	0-4-1	0-5-0	0-2-3	-	0-11-4

From the hypothesis tests the EMO algorithms can be ranked for each performance metric over all five data sets. For the HV metric, the MOPSO and NSGA-III algorithms performed the best with ten wins each; the MO-CMA-ES algorithm was third (six wins); and finally the MOEAD (one win). For the SM and NPS metrics the order of best performing algorithms was similar. The best performing algorithm for the SM metric was the MOEAD (eight wins), followed by MO-CMA-ES (two wins), MOPSO (zero wins), and lastly, the NSGA-III (zero wins). The NPS metric had similar results, the MOEAD (nine wins), followed by MO-CMA-ES (two wins), MOPSO (zero wins), and lastly the NSGA-III (zero wins). These results indicate that if accuracy is the most important concern, MOPSO should be the algorithm of choice and if spread and number of solutions are more important, the MOEAD algorithm should be used in the EMO algorithm framework.

9.6 Summary

The aim of this section is to propose an evolutionary multi-objective algorithm framework for scheduling picking and binning robots in an automated warehouse. The framework was tested using four popular EMO algorithms; namely, NSGA-III, MOEAD, MOPSO, and MO-CMA-ES. Five real-world data sets corresponding to different demand scenarios were used for evaluation. The framework was able to produce feasible trade-off solutions and MOPSO was shown to be the best EMO algorithm to use for accuracy. If a large spread and number of Pareto solutions are the most important concern, MOEAD should be used.

Chapter 10

Conclusion

This chapter summarises the main findings of the dissertation's single objective part sequencing and allocation problem as well as the multi-objective part sequencing and allocation problem. Possible future research opportunities are also identified and discussed in this chapter.

10.1 Single objective part sequencing and allocation problem summary

The first aim of this dissertation is to develop a framework for solving the robot routing and single objective part sequencing and allocation problem. Solving the single objective part sequencing and allocation problem requires the subproblem, the robot routing problem, to be solved first. The robot routing solution is used to navigate the robot from point A to a point B using a set of rules. The routing solution needs to be as efficient as possible to minimise the total travelling time.

The solution strategies for the robot routing problem consist of the return heuristic, the s-shape heuristic, the midpoint heuristic and the largest gap heuristic. The four routing heuristics are analysed for 30 random sequences over five data sets. The results were obtained and the return heuristic outperformed the s-shape heuristic, the midpoint heuristic and the largest gap heuristic.

After the best routing heuristic was determined, a framework was developed for solving the single objective part sequencing and allocation problem. All the parts had to be picked and binned, and allocated to a robot in order to solve the single objective part sequencing and allocation problem. Four metaheuristics are tested within the framework to solve the single objective part sequencing and allocation problem; namely, the covariance matrix adaptation evolution strategy (CMA-ES) algorithm, the genetic algorithm (GA), the guaranteed convergence particle swarm optimisation (GCPSO) algorithm, and the self-adaptive differential evolution algorithm with neighbourhood search (SaNSDE). The framework is responsible for optimising the sequence in which the binning and picking takes place as well as the allocation to a robot. The four metaheuristics are tested on five data sets based on real-world data, with 8, 16, 24, 32, and 40 parts, respectively. Each experiment over the five data sets consists of thirty random independent runs using the four metaheuristics.

After this the experimental runs were completed the results were obtained and analysed. The results included the fitness function values and the population diversity values. The fitness function values gave valuable insight into the performance of each metaheuristic. It was concluded, statistically, that the CMA-ES statistically outperformed the other metaheuristics in solving the single objective part sequencing and allocation problem and was thus the best metaheuristic to use within the framework to solve this specific version of the algorithm.

The population diversity value is an indication of the spread of the population through the search space. It was positive to note that for all the metaheuristics over all the data sets, the diversity value never decreased to zero, indicating that an acceptable exploration-exploitation balance was maintained for all algorithms.

The results obtained from the single objective framework are a valuable contribution to literature as this is the first time that the four single objective evolutionary algorithms have been tested in solving the single objective part sequencing and allocation problem in a warehouse environment with the incorporation of collision avoidance.

After it was determined that the CMA-ES was the best algorithm to use when solving the single objective part sequencing and allocation problem, a sensitivity analysis was conducted. The algorithm was tested to see what effect the number of robots might have on the solution. The results were positive and showed that an increase in robots leads to a decrease in fitness function value. The algorithm showed an improvement in fitness function value to a limit of seven robots.

Overall, the framework showed that the CMA-ES, along with the return routing heuristic, was effective and applicable in generating feasible solutions and optimising the single objective part sequencing and allocation problem.

10.2 Multi-objective part sequencing and allocation problem summary

The second aim of this dissertation is to develop an EMO framework in order to solve the multi-objective part sequencing and allocation problem. Solving the multi-objective part sequencing and allocation problem also requires the best routing heuristic to navigate the robot through the warehouse.

The same routing heuristic is used for the multi-objective part sequencing and allocation problem as for the single objective part sequencing and allocation problem; namely, the return heuristic. All the parts had to be picked and binned, and allocated to a robot in order to solve the multi-objective part sequencing and allocation problem. Four EMO algorithms are used to test the EMO framework on the multi-objective part sequencing and allocation problem; namely, the MO-CMA-ES, the NSGA-III, MOPSO and the MOEAD algorithms. The framework is again responsible for optimising the sequence in which the binning and picking takes place as well as the allocation to a robot. The four algorithms are tested on five data sets based on real-world data, with 8, 16, 24, 32, and 40 parts, respectively. Each experiment over the five data sets consists of thirty random independent runs using the four metaheuristics. The framework could generate feasible solutions and optimise the multi-objective part sequencing and allocation problem.

After the empirical evaluation the results were obtained and analysed. The results included three metrics; namely, the hypervolume, the spread metric and the number of Pareto solutions metric. From the iteration average results and hypothesis test results for each metric, the ranking for the EMO algorithms could be confirmed. For the hypervolume metric the MOPSO

and NSGA-III algorithms performed the best. The spread metric and number of Pareto solutions metric had the same order of best performing algorithms. The best performing algorithm was the MOEAD (eight wins), followed by MO-CMA-ES (two wins), MOPSO (zero wins), and lastly the NSGA-III (zero wins). To conclude the description of the multi-objective part sequencing and allocation problem, the results confirmed that if accuracy is the highest priority, MOPSO should be the algorithm of choice and if spread and number of solutions are more important, then the MOEAD algorithm should be used in the EMO algorithm framework.

The results from the multi-objective framework are a valuable contribution to literature as this is the first time that the EMO framework with four respective multi-objective evolutionary algorithms has been tested on solving the multi-objective part sequencing and allocation problem in a warehouse environment with the incorporation of collision avoidance.

10.3 Future research opportunities

A number of opportunities for future research exist and are described in detail in this section.

1) Introducing stochastic variables:

Currently all time variables are treated as constant variables. In a real-world scenario, as in the case of this problem, travelling times, picking times, and binning times are stochastic in nature. The algorithm can be improved to accommodate real-world time instances using stochastic distributions obtained from time studies and previous data.

2) Allow batching of parts:

The current algorithm does not allow for any parts to be batched together when picking and binning. Introducing the option to batch parts together will definitely improve the total picking and binning time. The batching of parts will, however, lead to a drastic increase in the number of decision variables and the computational power required to solve the problem.

3) Real-time dynamic optimisation:

The problem can also be formulated and solved as a dynamic robot routing problem, where decisions regarding to routing, sequencing or allocation need to be made in real-time due to environmental changes.

4) Investigating differentiating problem characteristics:

It was clear in the results presented in this dissertation that no single algorithm is capable of outperforming all other algorithms for all performance measurements over all data sets. An investigation into the problem characteristics which drive algorithm performance might indicate which algorithms are more suitable for solving each specific part sequencing and allocation problem data set.

5) Warehouse layout design improvement:

This dissertation touched on the subject of layout design. An investigation into the effect of different warehouse layout designs with the same data sets could prove to have a significant impact on the solutions found for the routing, part sequencing, and allocation problems.

6) Parameter tuning:

Parameters of test algorithms can be tuned and optimised to perform optimally within the developed framework.

7) Alternative problem representations:

Alternative problem representations and mapping mechanisms for converting a set of decision variables into a robot sequencing and allocation problem could be investigated.

10.4 Last words

Given the increase in the number of warehouses and their sizes for various industries, companies are aiming to optimise their warehouse activities to be as efficient as possible. Most supply chains in South Africa make use of warehousing facilities. The sequencing and allocation algorithm framework developed from this research can be utilised in any warehouse environment where a picking and binning process is present. The research contribution of this work includes the development of a framework which incorporated collision avoidance to the single objective and multi-objective part sequencing and allocation problem. The research contributes to the field of metaheuristics where the four metaheuristics namely the CMA-ES, GA, GCPSO, and SaNSDE, have been tested for solving the robot picking and binning problem. The research also contributes to the field of EMO algorithms where the four EMO algorithms; namely, the MO-CMA-ES, NSGA-III, MOPSO and MOEAD, have been tested for solving the multi-objective part sequencing and allocation problem. Given the imminence of the fourth industrial revolution, companies operating warehouses can benefit immensely from automation research into robotic warehouse picking.

Bibliography

Andelković, A. and Radosavljević, M. (2018). Improving order-picking process through implementation of warehouse management system. *Strategic Management*, 23(1):3–10.

Auger, A. and Hansen, N. (2005). A restart cma evolution strategy with increasing population size. In *2005 IEEE congress on evolutionary computation*, volume 2, pages 1769–1776. IEEE.

Azadeh, A., Shafiee, F., Yazdanparast, R., Heydari, J., and Fathabad, A. M. (2017). Evolutionary multi-objective optimization of environmental indicators of integrated crude oil supply chain under uncertainty. *Journal of cleaner production*, 152:295—311.

Baker, P. (2004). The adoption of innovative warehouse equipment. In *Logistics Research Network 2004 Conference Proceedings*, pages 25–35.

Baker, P. and Canessa, M. (2009). Warehouse design: A structured approach. *European Journal of Operational Research*, 193(2):425–436.

Baker, P. and Halim, Z. (2007). An exploration of warehouse automation implementations: cost, service and flexibility issues. *Supply Chain Management: An International Journal*, 12(2):129–138.

Banker, S. (2009). Warehouse 2025. *ARC Advisory Group, Tech. Rep.*

Bechtsis, D., Tsolakis, N., Vlachos, D., and Iakovou, E. (2017). Sustainable supply chain management in the digitalisation era: The impact of automated guided vehicles. *Journal of Cleaner Production*, 142:3970–3984.

Beroule, B., Grunder, O., Barakat, O., and Aujoulat, O. (2017). Order picking problem in a warehouse hospital pharmacy. *IFAC-PapersOnLine*, 50(1):5017–5022.

Biswas, S., Anavatti, S. G., and Garratt, M. A. (2021). Multiobjective mission route planning problem: A neural network-based forecasting model for mission planning. *IEEE Transactions on Intelligent Transportation Systems*, 1:430–422.

Broulias, G., Marcoulaki, E., Chondrocoukis, G., and Laios, L. (2005). Warehouse management for improved order picking performance: an application case study from the wood industry. *Department of Industrial Management & Technology, University of Piraeus.*

Brownlee, J. (2011). *Clever algorithms: nature-inspired programming recipes.* Jason Brownlee.

Burkard, R. E., Deineko, V. G., van Dal, R., van der Veen, J. A., and Woeginger, G. J. (1998). Well-solvable special cases of the traveling salesman problem: a survey. *SIAM review*, 40(3):496–546.

Cesarone, J. and Eman, K. F. (1989). Mobile robot routing with dynamic programming. *Journal of Manufacturing Systems*, 8(4):257–266.

Chen, G., Hou, J., Dong, J., Li, Z., Gu, S., Zhang, B., Yu, J., and Knoll, A. (2020). Multi-objective scheduling strategy with genetic algorithm and time enhanced a* planning for autonomous parking robotics in high-density unmanned parking lots. *IEEE/ASME Transactions on Mechatronics.*

Chen, G. and Liu, J. (2019). Mobile robot path planning using ant colony algorithm and improved potential field method. *Computational intelligence and neuroscience*, 2019.

Chen, H., Wang, Q., Yu, M., Cao, J., and Sun, J. (2018). Path planning for multi-robot systems in intelligent warehouse. In *International Conference on Internet and Distributed Computing Systems*, pages 148–159. Springer.

Christofides, N. (1975). *Graph theory: An algorithmic approach (Computer science and applied mathematics)*. Academic Press, Inc.

Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. (2007). *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer.

Coello, C. C. and Lechuga, M. S. (2002). Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1051–1056. IEEE.

Croucamp, M. and Grobler, J. (2021). Metaheuristics for the robot part sequencing and allocation problem with collision avoidance. In *EPIA Conference on Artificial Intelligence*, pages 469–481. Springer.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410.

Dao, T.-K., Pan, T.-S., and Pan, J.-S. (2016). A multi-objective optimal mobile robot path planning based on whale optimization algorithm. In *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, pages 337–342. IEEE.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007a). Design and control of warehouse order picking: A literature review. *European journal of operational research*, 182(2):481–501.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007b). Design and control of warehouse order picking: A literature review. *European journal of operational research*, 182(2):481–501.

De Ryck, M., Versteyhe, M., and Debrouwere, F. (2020). Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, 54:152–173.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International conference on parallel problem solving from nature*, pages 849–858. Springer.

Deb, K. and Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Dekker, R., De Koster, M., Roodbergen, K. J., and Van Kalleveen, H. (2004). Improving order-picking response time at ankor's warehouse. *Interfaces*, 34(4):303–313.

Dhawale, A., Yang, X., and Michael, N. (2018). Reactive collision avoidance using real-time local gaussian mixture model maps. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3545–3550. IEEE.

Digani, V., Caramaschi, F., Sabattini, L., Secchi, C., and Fantuzzi, C. (2014). Obstacle avoidance for industrial agvs. In *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 227–232. IEEE.

Digani, V., Hsieh, M. A., Sabattini, L., and Secchi, C. (2019). Coordination of multiple agvs: a quadratic optimization method. *Autonomous Robots*, 43(3):539–555.

Dong, Y.-f., Xia, H.-m., and Zhou, Y.-c. (2016). Disordered and multiple destinations path planning methods for mobile robot in dynamic environment. *Journal of Electrical and Computer Engineering*, 2016.

Draganjac, I., Petrović, T., Miklić, D., Kovačić, Z., and Oršulić, J. (2020). Highly-scalable traffic management of autonomous industrial transportation systems. *Robotics and Computer-Integrated Manufacturing*, 63:101915.

Dukic, G. and Oluic, C. (2007). Order-picking methods: improving order-picking efficiency. *International Journal of Logistics Systems and Management*, 3(4):451–460.

Eberhart, R. and Kennedy, J. (1995a). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee.

Eberhart, R. and Kennedy, J. (1995b). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer.

Engelbrecht, A. (2006). Fundamentals of computational swarm intelligence. *Hoboken: John Wiley & Sons, Ltd.*

Eshelman, L. J. and Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. *Foundations of genetic algorithms*, 2:187–202.

Fanti, M. P., Mangini, A. M., Pedroncelli, G., and Ukovich, W. (2015). Decentralized deadlock-free control for agv systems. In *2015 American Control Conference (ACC)*, pages 2414–2419. IEEE.

Fazlollahtabar, H. and Hassanli, S. (2018). Hybrid cost and time path planning for multiple autonomous guided vehicles. *Applied Intelligence*, 48(2):482–498.

Fonlupt, J. and Naddef, D. (1992). The traveling salesman problem in graphs with some excluded minors. *Mathematical Programming*, 53(1-3):147–172.

Ganapathy, V., Jie, T. T. J., and Parasuraman, S. (2010). Improved ant colony optimization for robot navigation. In *7th International Symposium on Mechatronics and its Applications*, pages 1–6. IEEE.

Gilmore, P. C. (1985). Well-solved special cases. *The traveling salesman problem*, pages 136–138.

Gochev, I., Nadzinski, G., and Stankovski, M. (2017). Path planning and collision avoidance regime for a multi-agent system in industrial robotics. *Machines. Technologies. Materials.*, 11(11):519–522.

Golberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989(102):36.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

Got, A., Moussaoui, A., and Zouache, D. (2020). A guided population archive whale optimization algorithm for solving multiobjective optimization problems. *Expert Systems with Applications*, 141:112972.

Grobler, J. (2015). *The heterogeneous meta-hyper-heuristic: from low level heuristics to low level meta-heuristics.* PhD thesis, University of Pretoria.

Grobler, J. et al. (2008). *Particle swarm optimization and differential evolution for multi-objective multiple machine scheduling.* PhD thesis, University of Pretoria.

Gu, Q., Wang, Q., Li, X., and Li, X. (2021). A surrogate-assisted multi-objective particle swarm optimization of expensive constrained combinatorial optimization problems. *Knowledge-Based Systems*, 223:107049.

Gue, K. R., Ivanović, G., and Meller, R. D. (2012). A unit-load warehouse with multiple pickup and deposit points and non-traditional aisles. *Transportation Research Part E: Logistics and Transportation Review*, 48(4):795–806.

Gue, K. R. and Meller, R. D. (2009). Aisle configurations for unit-load warehouses. *IIE transactions*, 41(3):171–182.

Hadka, D. (2015). Platypus.

Haiming, L., Weidong, L., Mei, Z., and An, C. (2019). Algorithm of path planning based on time window for multiple mobile robots in warehousing system. In *2019 Chinese Control Conference (CCC)*, pages 2193–2199. IEEE.

Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE transactions*, 25(4):76–87.

Han, M.-H., McGinnis, L. F., Shieh, J. S., and White, J. A. (1987). On sequencing retrievals in an automated storage/retrieval system. *IIE transactions*, 19(1):56–66.

Hansen, N. and Kern, S. (2004). Evaluating the cma evolution strategy on multimodal test functions. In *PPSN*, volume 8, pages 282–291. Springer.

Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18.

Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE.

Hassan-Pour, H., Mosadegh-Khah, M., and Tavakkoli-Moghaddam, R. (2009). Solving a multi-objective multi-depot stochastic location-routing problem by a hybrid simulated annealing algorithm. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 223(8):1045–1054.

Hennes, D., Claes, D., Meeussen, W., and Tuyls, K. (2012). Multi-robot collision avoidance with localization uncertainty. In *AAMAS*, pages 147–154.

Heragu, S. S. (2008). *Facilities design*. Crc Press.

Heris, M. K. (2015a). Moea/d in matlab.

Heris, M. K. (2015b). Multi-objective pso in matlab.

Heris, M. K. (2016). Nsga-iii: Non-dominated sorting genetic algorithm, the third version — matlab implementation.

Igel, C., Hansen, N., and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation*, 15(1):1–28.

Jabbarpour, M. R., Zarrabi, H., Jung, J. J., and Kim, P. (2017). A green ant-based method for path planning of unmanned ground vehicles. *IEEE access*, 5:1820–1832.

Jacobus, C. J., Beach, G. J., and Rowe, S. (2015). Automated warehousing using robotic forklifts. US Patent 8,965,561.

Jain, H. and Deb, K. (2013). An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 307–321. Springer.

Jan, G. E., Chang, K. Y., and Parberry, I. (2008). Optimal path planning for mobile robot navigation. *IEEE/ASME transactions on mechatronics*, 13(4):451–460.

Juntao, L., Tingting, D., Yuanyuan, L., and Yan, H. (2016). Study on robot path collision avoidance planning based on the improved ant colony algorithm. In *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 2, pages 540–544. IEEE.

Karásek, J. (2013). An overview of warehouse optimization. *International journal of advances in telecommunications, electrotechnics, signals and systems*, 2(3):111–117.

Khamis, A., Hussein, A., and Elmogy, A. (2015). Multi-robot task allocation: A review of the state-of-the-art. *Cooperative Robots and Sensor Networks 2015*, pages 31–51.

Kim, B.-I., Heragu, S. S., Graves, R. J., and Onge, A. S. (2003). Clustering-based order-picking sequence algorithm for an automated warehouse. *International Journal of Production Research*, 41(15):3445–3460.

Klaas, A., Laroque, C., Dangelmaier, W., and Fischer, M. (2011). Simulation aided, knowledge based routing for agvs in a distribution warehouse. *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 1668–1679.

Kleyn, R. (2020). Investigating the conflict in conflicting objectives in the scheduling of picking and binning robots within a warehouse. University of Stellenbosch. Final year project.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2):149–172.

Kovács, S. and Kóczy, L. T. (1999). Application of an approximate fuzzy logic controller in an agv steering system, path tracking and collision avoidance strategy. *Fuzzy Set Theory and Applications, In Tatra Mountains Mathematical Publications, Mathematical Institute Slovak Academy of Sciences*, 16:456–467.

Kulatunga, A., Liu, D., Dissanayake, G., and Siyambalapitiya, S. (2006). Ant colony optimization based simultaneous task allocation and path planning of autonomous vehicles. In *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–6. IEEE.

Larsen, L., Kim, J., Kupke, M., and Schuster, A. (2017). Automatic path planning of industrial robots comparing sampling-based and computational intelligence methods. *Procedia Manufacturing*, 11:241–248.

Lee, H.-Y., Shin, H., and Chae, J. (2018). Path planning for mobile agents using a genetic algorithm with a direction guided factor. *Electronics*, 7(10):212.

Lee, P.-S. and Wang, L.-L. (1994). Collision avoidance by fuzzy logic control for automated guided vehicle navigation. *Journal of robotic systems*, 11(8):743–760.

Li, J., Huang, R., and Dai, J. B. (2017). Joint optimisation of order batching and picker routing in the online retailer's warehouse in china. *International Journal of Production Research*, 55(2):447–461.

Li, X., Zhang, C., Yang, W., and Qi, M. (2018). Multi-agvs conflict-free routing and dynamic dispatching strategies for automated warehouses. In *International Conference on Mobile and Wireless Technology*, pages 277–286. Springer.

Liu, D. and Kulatunga, A. (2007). Simultaneous planning and scheduling for multi-autonomous vehicles. In *Evolutionary Scheduling*, pages 437–464. Springer.

Liu, Y., Wang, L., Huang, H., Liu, M., and Xu, C.-z. (2017). A novel swarm robot simulation platform for warehousing logistics. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2669–2674. IEEE.

López-González, A., Campaña, J. M., Martínez, E. H., and Contro, P. P. (2020). Multi robot distance based formation using parallel genetic algorithm. *Applied Soft Computing*, 86:105929.

Majumder, A. and Ghosh, R. (2020). Task allocation and path planning of a multi-robot system using heuristic coupled particle swarm optimization algorithm. In *Handbook of Research on Developments and Trends in Industrial and Materials Engineering*, pages 194–209. IGI Global.

Majumder, A., Majumder, A., and Bhaumik, R. (2021). Teaching–learning-based optimization algorithm for path planning and task allocation in multi-robot plant inspection system. *Arabian Journal for Science and Engineering*, pages 1–23.

Menon, S., Kapoor, S. G., and Blackmon, R. (1988). Navigation planning for mobile robotic devices in modular warehouses. *The International Journal of Advanced Manufacturing Technology*, 3(4):47–62.

Mitchell, M. (1996). An introduction to genetic algorithms mit press. *Cambridge, Massachusetts. London, England.*

Moeller, K. (2011). Increasing warehouse order picking performance by sequence optimization. *Procedia-Social and Behavioral Sciences*, 20:177–185.

Möhring, R. H., Köhler, E., Gawrilow, E., and Stenzel, B. (2005). Conflict-free real-time agv routing. In *Operations Research Proceedings 2004*, pages 18–24. Springer.

Molnar, B. and Lipovszki, G. (2005). Multi-objective routing and scheduling of order pickers in a warehouse. *International Journal of Simulation*, 6(5):22–32.

Ó Duinn, J. (1994). *Robotic navigation.* PhD thesis, Dublin City University.

Okumuş, F., Dönmez, E., and Kocamaz, A. F. (2020). A cloudware architecture for collaboration of multiple agvs in indoor logistics: Case study in fabric manufacturing enterprises. *Electronics*, 9(12):2023.

Olorunda, O. and Engelbrecht, A. P. (2009). An analysis of heterogeneous cooperative algorithms. *2009 IEEE Congress on Evolutionary Computation*, pages 1562–1569.

Ozden, S. G. (2017). A computational system to solve the warehouse aisle design problem. *A dissertation submitted to the Graduate Faculty of Auburn University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.*

Pal, N. S. and Sharma, S. (2013). Robot path planning using swarm intelligence: A survey. *International Journal of Computer Applications*, 83(12):5–12.

Pamosoaji, A. K. and Hong, K.-S. (2011). Collision-free path and trajectory planning algorithm for multiple-vehicle systems. In *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, pages 67–72. IEEE.

Qin, A. K. and Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. *2005 IEEE congress on evolutionary computation*, 2:1785–1791.

Rardin, R. L. and Rardin, R. L. (1998). *Optimization in operations research*, volume 166. Prentice Hall Upper Saddle River, NJ.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521.

Roodbergen, K. J. and De Koster, R. (2001). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43.

Roodbergen, K. J. and Koster, R. (2001). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883.

Ros, R. and Hansen, N. (2008). A simple modification in cma-es achieving linear time and space complexity. In *International Conference on Parallel Problem Solving from Nature*, pages 296–305. Springer.

Sadeghi, J., Sadeghi, S., and Niaki, S. T. A. (2014). Optimizing a hybrid vendor-managed inventory and transportation problem with fuzzy demand: an improved particle swarm optimization algorithm. *Information Sciences*, 272:126–144.

Sahoo, R. R., Rakshit, P., Haidar, M. T., Swarnalipi, S., Balabantaray, B., and Mohapatra, S. (2011). Navigational path planning of multi-robot using honey bee mating optimization algorithm (hbmo). *International Journal of Computer Applications*, 27(11):1–8.

Schulze, L., Behling, S., and Buhrs, S. (2008). Automated guided vehicle systems: a driver for increased business performance. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 2, pages 1–6.

Shir, O. M., Emmerich, M., and Bäck, T. (2010). Adaptive niche radii and niche shapes approaches for niching with the cma-es. *Evolutionary computation*, 18(1):97–126.

Stoean, C., Preuss, M., Stoean, R., and Dumitrescu, D. (2010). Multimodal optimization by means of a topological species conservation algorithm. *IEEE Transactions on Evolutionary Computation*, 14(6):842–864.

Storn, R. (1996). On the usage of differential evolution for function optimization. In *Proceedings of North American Fuzzy Information Processing*, pages 519–523. IEEE.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Suchi, M. M. and Vincze, M. (2014). Meta-heuristic search strategies for local path-planning to find collision free trajectories. In *Proceedings of the Austrian Robotics Workshop (ARW-14)*, pages 36–41.

Sullivan, N., Grainger, S., and Cazzolato, B. (2018). A dual genetic algorithm for multi-robot routing with network connectivity and energy efficiency. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1647–1652. IEEE.

Tan, C. S. (2006). A collision avoidance system for autonomous underwater vehicles. *Research Theses Main Collection*.

Theys, C., Bräysy, O., Dullaert, W., and Raa, B. (2007). Towards a metaheuristic for routing order pickers in a warehouse. *Evolutionary methods for design, optimization and control*, pages 385–390.

Tompkins, J., White, J., Bozer, Y., Frazelle, E., and Tanchoco, J. (1996). Trevino. *Facilities planning*, 9.

Truong, N. C., Dang, T. G., and Nguyen, D. A. (2018). Optimizing automated storage and retrieval algorithm in cold warehouse by combining dynamic routing and continuous cluster method. In *International Conference on Advanced Engineering Theory and Applications*, pages 283–293. Springer.

Uriol, R. and Moran, A. (2017). Mobile robot path planning in complex environments using ant colony optimization algorithm. In *2017 3rd international conference on control, automation and robotics (ICCAR)*, pages 15–21. IEEE.

Van den Bergh, F. and Engelbrecht, A. P. (2002). A new locally convergent particle swarm optimiser. In *IEEE International conference on systems, man and cybernetics*, volume 3, pages 6–pp. IEEE.

Vanderpooten, D. (1990). L'approche interactive dans l'aide multicritère à la décision. *These de doctorat, Université de Paris IX-Dauphine, France*.

Vesterstrom, J. S., Riget, J., and Krink, T. (2002). Division of labor in particle swarm optimisation. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1570–1575. IEEE.

Vis, I. F., De Koster, R., Roodbergen, K. J., and Peeters, L. W. (2001). Determination of the number of automated guided vehicles required at a semi-automated container terminal. *Journal of the Operational research Society*, 52(4):409–417.

Vivaldini, K., Rocha, L. F., Martarelli, N. J., Becker, M., and Moreira, A. P. (2016). Integrated tasks assignment and routing for the estimation of the optimal number of agvs. *The International Journal of Advanced Manufacturing Technology*, 82(1-4):719–736.

Vivaldini, K. C., Galdames, J. P., Bueno, T. S., Araújo, R. C., Sobral, R. M., Becker, M., and Caurin, G. A. (2010). Robotic forklifts for intelligent warehouses: Routing, path planning, and auto-localization. In *Industrial Technology (ICIT), 2010 IEEE International Conference on*, pages 1463–1468. IEEE.

Vivaldini, K. C. T., Becker, M., and Caurin, G. A. (2009). Automatic routing of forklift robots in warehouse applications. In *Proceedings of the 20th International Congress of Mechanical Engineering, Gramado-RS, Brazil*. Citeseer.

Wang, C.-Y. and Banitaan, S. (2018). A partitioning-based approach for robot path planning problems. In *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 178–182. IEEE.

Watanabe, M., Furukawa, M., and Kakazu, Y. (2001). Intelligent agv driving toward an autonomous decentralized manufacturing system. *Robotics and computer-integrated manufacturing*, 17(1-2):57–64.

Werner, F. (2006). V. t'kindt and jc billaut, multicriteria scheduling. theory, models and algorithms, springer verlag (2002) isbn 3-540-43617-0. *European Journal of Operational Research*, 168(1):275–277.

Wong, K.-C., Leung, K.-S., and Wong, M.-H. (2010). Protein structure prediction on a lattice model via multimodal optimization techniques. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 155–162.

Wu, K.-H., Chen, C.-H., Ko, J.-M., and Lee, J.-D. (1999). Path planning and prototype design of an agv. *Mathematical and computer modelling*, 30(7-8):147–167.

Wu, T.-H., Low, C., and Bai, J.-W. (2002). Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29(10):1393–1415.

Wyland, B. (2008). Warehouse automation: How to implement tomorrow's order fulfillment system today. *Aberdeen Group, Boston Google Scholar*.

Yan, X., Zhang, C., and Qi, M. (2017). Multi-agvs collision-avoidance and deadlock-control for item-to-human automated warehouse. In *2017 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)*, pages 1–5. IEEE.

Yang, Z., Tang, K., and Yao, X. (2008). Self-adaptive differential evolution with neighborhood search. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1110–1116. IEEE.

Yang, Z., Yao, X., and He, J. (2007). Making a difference to differential evolution. *Advances in metaheuristics for hard optimization*, pages 397–414.

Yuan, R., Dong, T., and Li, J. (2016). Research on the collision-free path planning of multi-agvs system based on improved a* algorithm. *American Journal of Operations Research*, 6(6):442–449.

Zhang, Q. and Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731.

Zhang, W., Peng, Y., Wei, W., and Kou, L. (2018a). Real-time conflict-free task assignment and path planning of multi-agv system in intelligent warehousing. In *2018 37th Chinese Control Conference (CCC)*, pages 5311–5316. IEEE.

Zhang, Z., Guo, Q., Chen, J., and Yuan, P. (2018b). Collision-free route planning for multiple agvs in an automated warehouse based on collision classification. *IEEE Access*, 6:26022–26035.

Zhang, Z., Guo, Q., and Yuan, P. (2017). Conflict-free route planning of automated guided vehicles based on conflict classification. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1459–1464. IEEE.

Zhuang, X., Feng, G., Lv, H., Lv, H., Wang, H., Zhang, L., Lin, J., and Tang, M. (2018). A collision-free path planning approach for multiple robots under warehouse scenarios. In *China Conference on Wireless Sensor Networks*, pages 55–63. Springer.

Zitzler, E., Deb, K., and Thiele, L. (1999). Comparison of multiobjective evolutionary algorithms: Empirical results (revised version). Technical report, Technical Report 70, Computer Engineering and Networks Laboratory (TIK ....

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271.

Appendices

Appendix A

The data sets were investigated to analyse the correlation between the different objective functions. From the histogram in Figure 65 to Figure 68, it can be seen that the increase in parts had a significant impact on the makespan, waiting time and number of collisions. It is interesting to note that idle time stayed constant rather than increasing according to the number of parts.

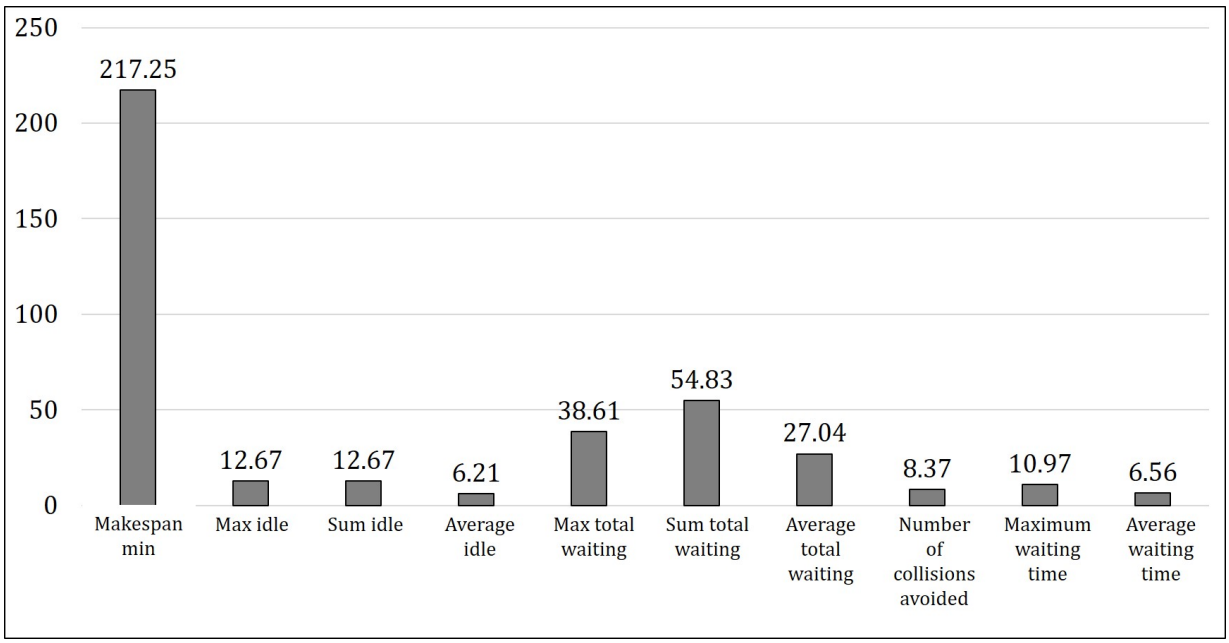

Figure 65: Average objective function values for 8 parts

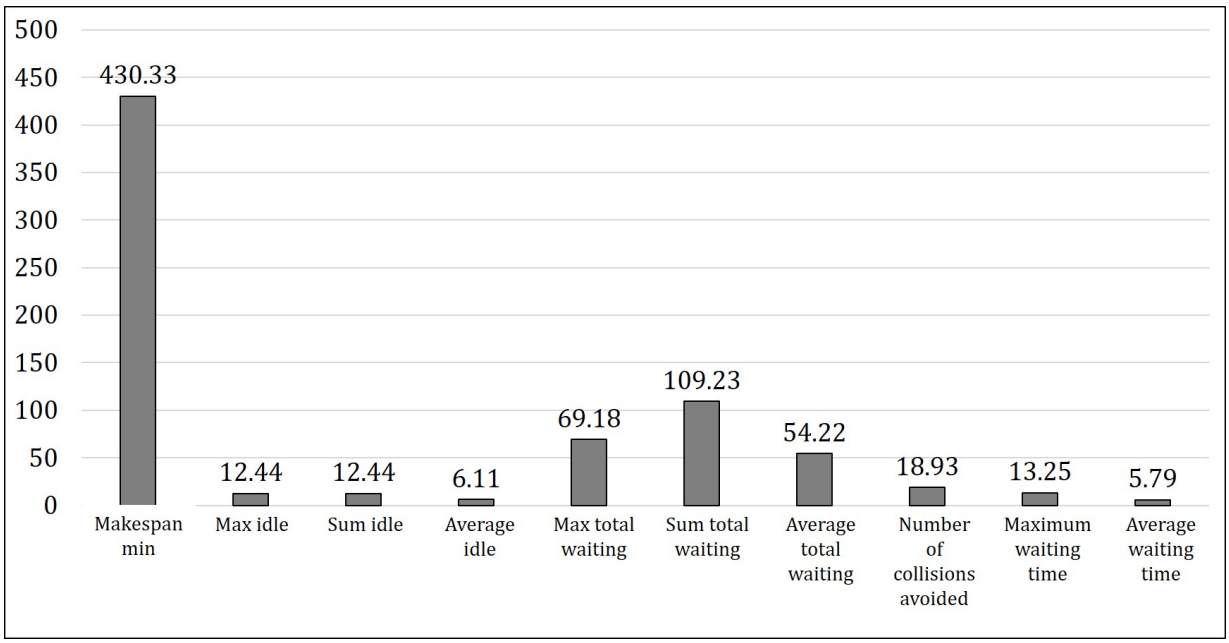

Figure 66: Average objective function values for 16 parts

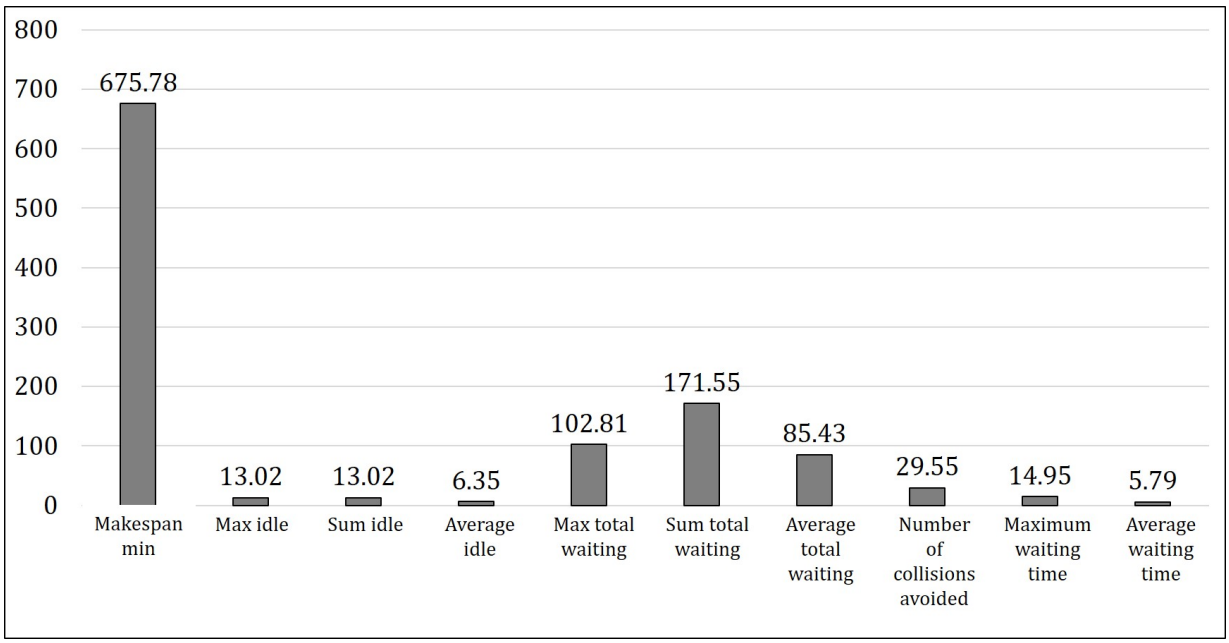

Figure 67: Average objective function values for 24 parts

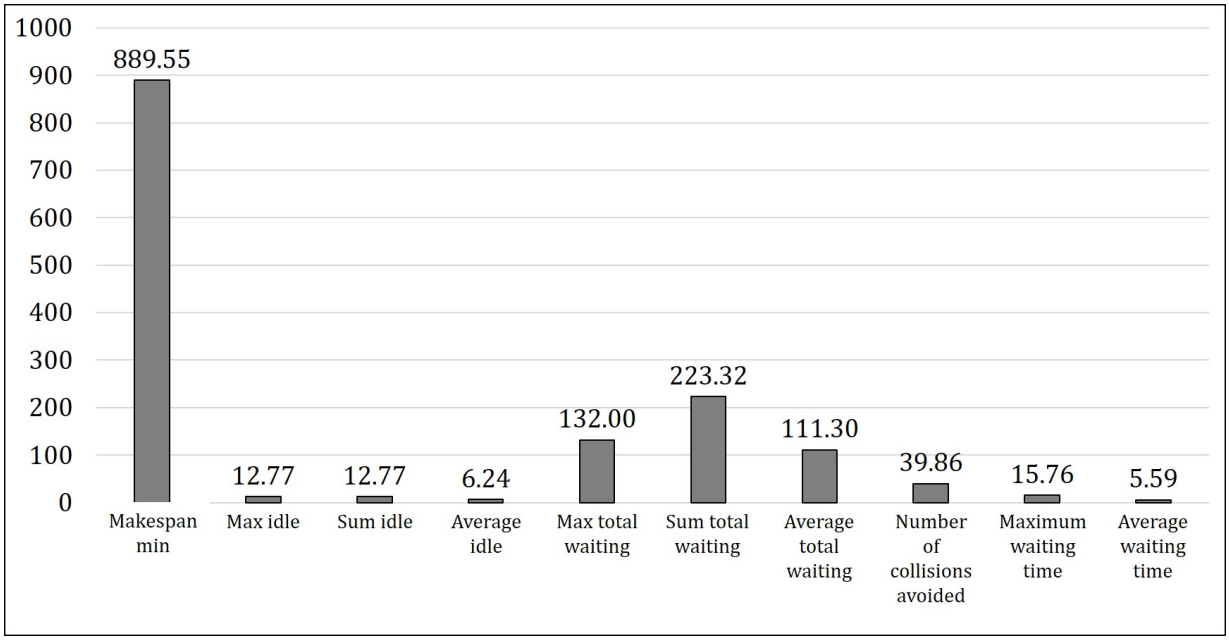

Figure 68: Average objective function values for 32 parts

Appendix B

The correlations between different objective functions can be seen from the SPLOM. It also provides an indication of whether the correlation was strong or weak. The correlation plot matrices for the for the smaller problems are presented in Figure 69 to Figure 72.

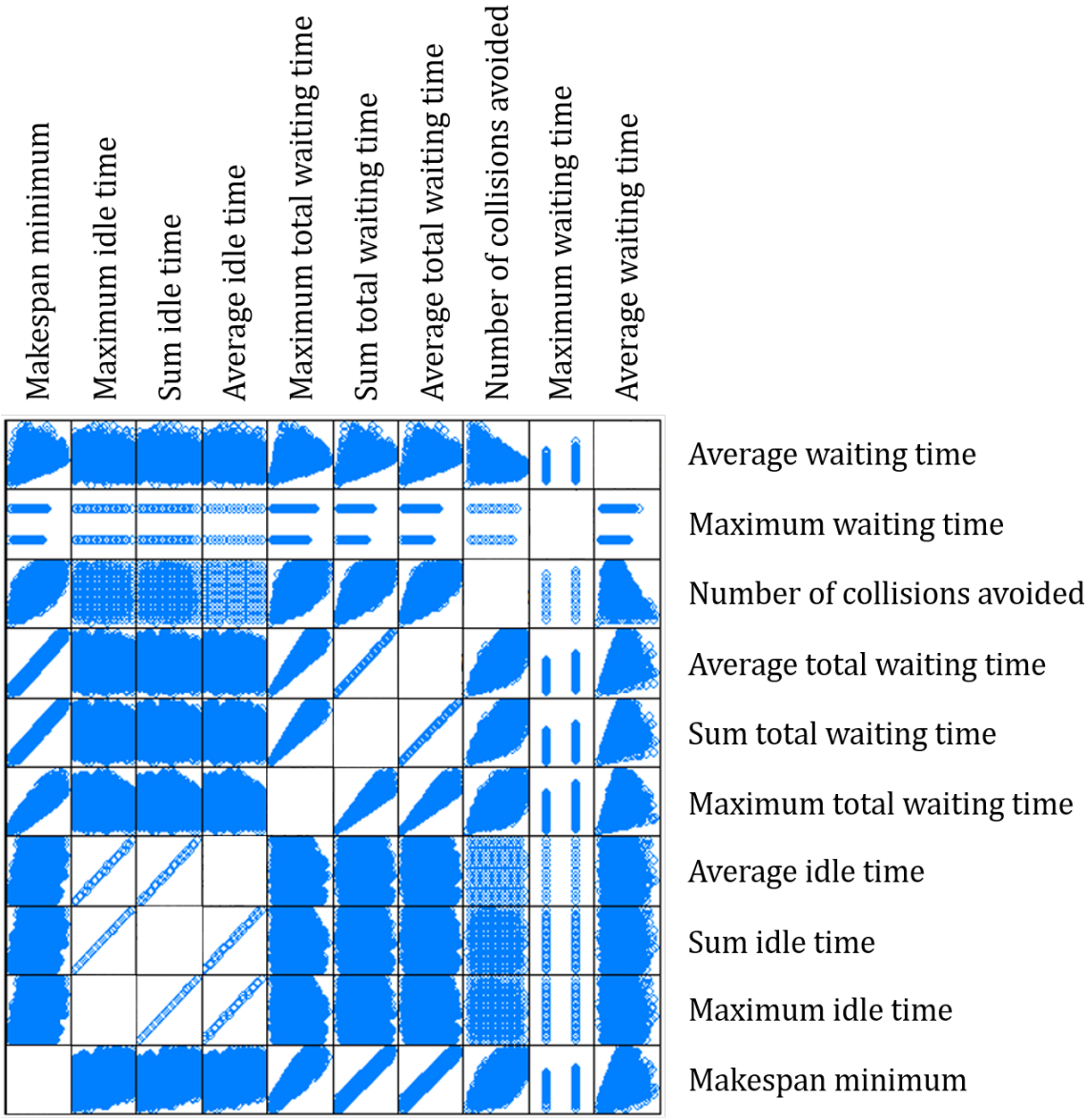

Figure 69: Correlation plot for 8 parts

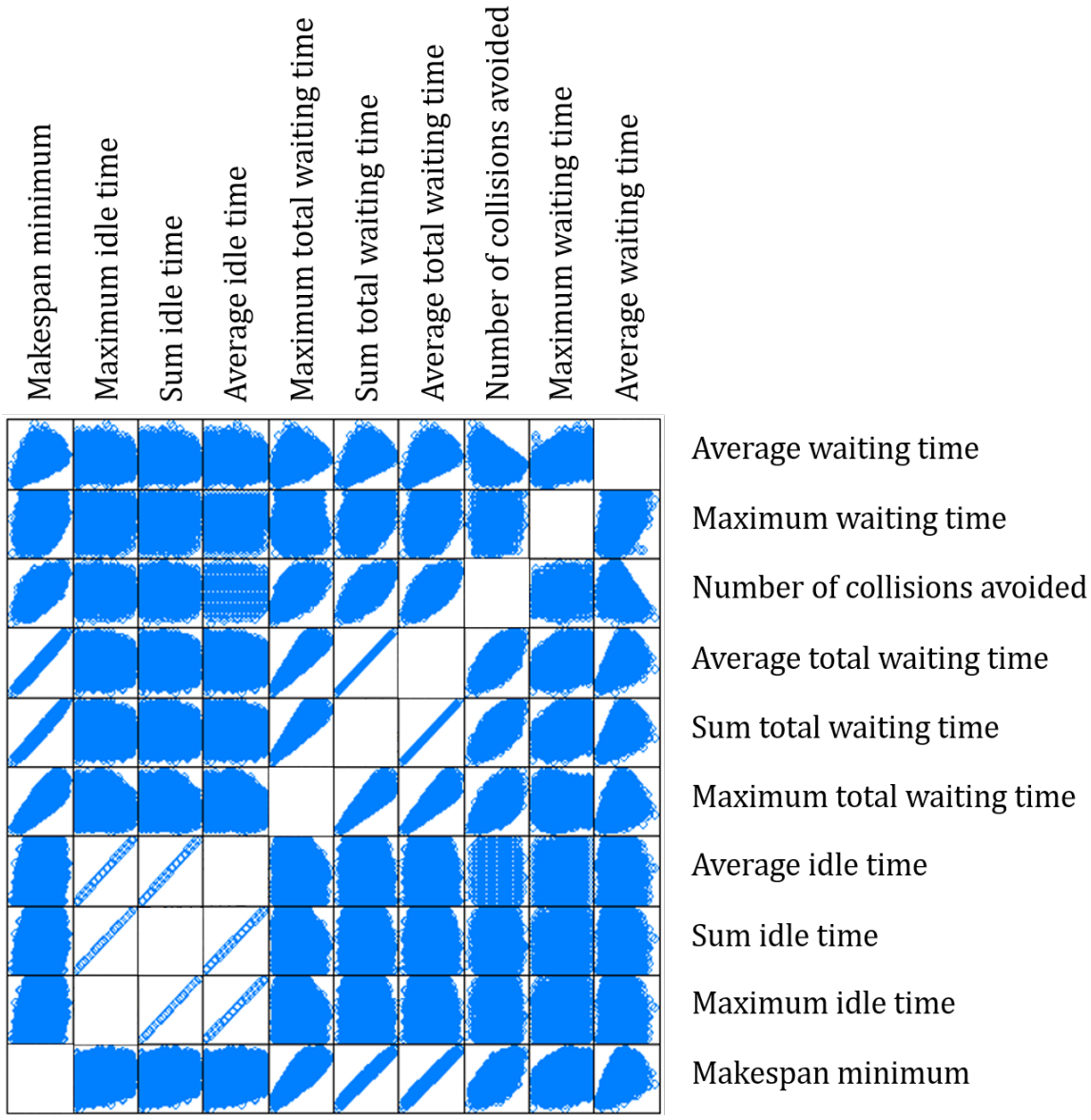

Figure 70: Correlation plot for 16 parts

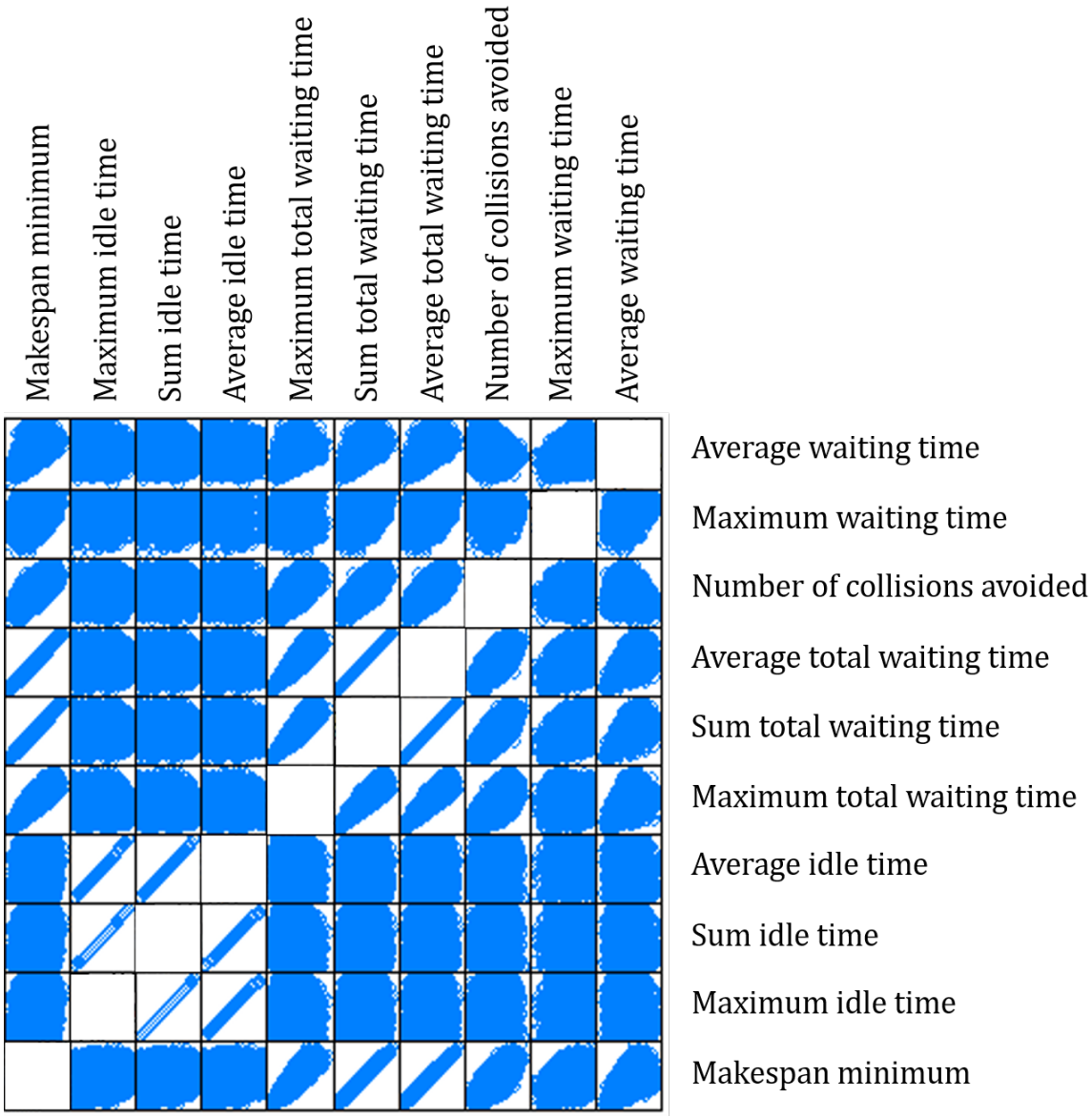

Figure 71: Correlation plot for 24 parts

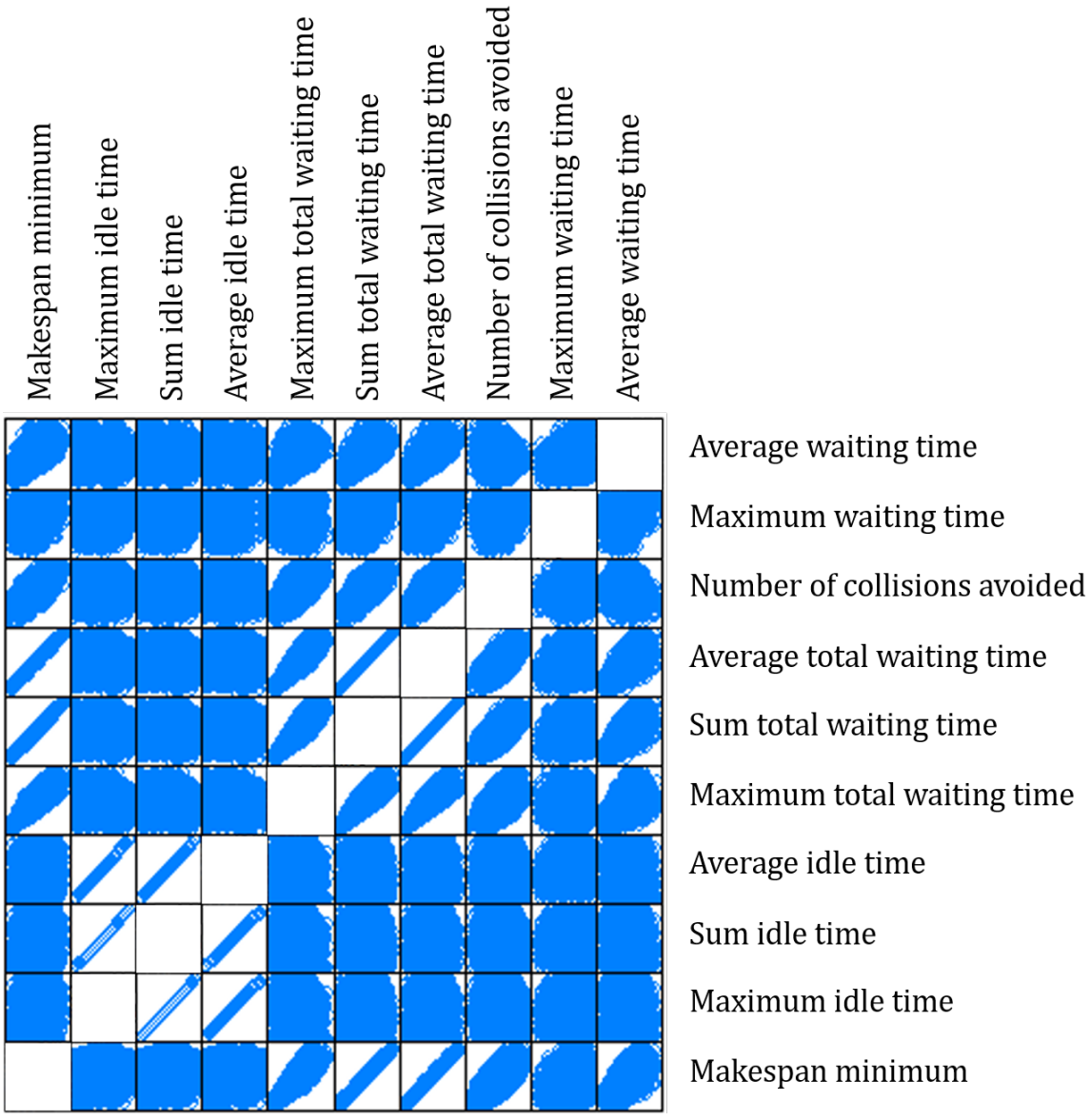

Figure 72: Correlation plot for 32 parts

Appendix C

The correlation coefficient tables for the four smaller problem sizes can be seen in Table 15 to Table 18.

Table 15: Correlation coefficients for 8 parts

	Maximum idle time	Sum idle time	Average idle time	Maximum total waiting time	Sum total waiting time	Average total waiting time	Number of collisions avoided	Maximum waiting time	Average waiting time
Makespan minimum	0.34	0.34	0.34	0.73	0.93	0.93	0.65	0.31	0.47
Maximum idle time		1.00	1.00	-0.08	-0.02	-0.02	-0.02	0.11	-0.01
Sum idle time			1.00	-0.08	-0.02	-0.02	-0.02	0.11	-0.01
Average idle time				-0.09	-0.03	-0.03	-0.02	0.10	-0.01
Maximum total waiting time					0.81	0.81	0.57	0.10	0.41
Sum total waiting time						1.00	0.70	0.29	0.50
Average total waiting time							0.70	0.29	0.50
Number of collisions avoided								0.10	-0.23
Maximum waiting time									0.30

Table 16: Correlation coefficients for 16 parts

	Maximum idle time	Sum idle time	Average idle time	Maximum total waiting time	Sum total waiting time	Average total waiting time	Number of collisions avoided	Maximum waiting time	Average waiting time
Makespan minimum	0.27	0.27	0.26	0.72	0.95	0.95	0.64	0.26	0.47
Maximum idle time		1.00	1.00	-0.05	-0.04	-0.04	-0.03	0.06	-0.01
Sum idle time			1.00	-0.05	-0.04	-0.04	-0.03	0.06	-0.01
Average idle time				-0.05	-0.04	-0.04	-0.03	0.06	-0.01
Maximum total waiting time					0.76	0.76	0.54	-0.02	0.34
Sum total waiting time						1.00	0.67	0.25	0.49
Average total waiting time							0.67	0.25	0.49
Number of collisions avoided								0.07	-0.30
Maximum waiting time									0.24

Table 17: Correlation coefficients for 24 parts

	Maximum idle time	Sum idle time	Average idle time	Maximum total waiting time	Sum total waiting time	Average total waiting time	Number of collisions avoided	Maximum waiting time	Average waiting time
Makespan minimum	0.21	0.21	0.21	0.77	0.97	0.97	0.67	0.25	0.58
Maximum idle time		1.00	1.00	-0.02	-0.01	-0.01	-0.01	0.05	-0.01
Sum idle time			1.00	-0.02	-0.01	-0.01	-0.01	0.05	-0.01
Average idle time				-0.02	-0.01	-0.02	0.00	0.04	-0.02
Maximum total waiting time					0.79	0.79	0.56	0.04	0.46
Sum total waiting time						1.00	0.69	0.25	0.59
Average total waiting time							0.69	0.25	0.59
Number of collisions avoided								0.10	-0.16
Maximum waiting time									0.23

Table 18: Correlation coefficients for 32 parts

	Maximum idle time	Sum idle time	Average idle time	Maximum total waiting time	Sum total waiting time	Average total waiting time	Number of collisions avoided	Maximum waiting time	Average waiting time
Makespan minimum	0.19	0.19	0.18	0.76	0.98	0.98	0.67	0.24	0.61
Maximum idle time		1.00	1.00	-0.02	-0.01	-0.01	-0.01	0.03	-0.01
Sum idle time			1.00	-0.02	-0.01	-0.01	-0.01	0.03	-0.01
Average idle time				-0.02	-0.01	-0.01	-0.01	0.03	-0.01
Maximum total waiting time					0.78	0.78	0.54	0.05	0.47
Sum total waiting time						1.00	0.68	0.24	0.62
Average total waiting time							0.68	0.24	0.62
Number of collisions avoided								0.10	-0.14
Maximum waiting time									0.22

Appendix D

This appendix lists the paper that has been published and the paper which is currently under review. The papers listed were derived from the work done in this dissertation.

- Croucamp M., Grobler J. (2021) Metaheuristics for the Robot Part Sequencing and Allocation Problem with Collision Avoidance. In: Marreiros G., Melo F.S., Lau N., Lopes Cardoso H., Reis L.P. (eds) Progress in Artificial Intelligence. EPIA 2021. Lecture Notes in Computer Science, vol 12981 (pp. 469-481). Springer, Cham.
- Croucamp, M. and Grobler, J. An evolutionary multi-objective optimisation algorithm framework for robot path planning, task allocation, and collision avoidance in an automated warehouse. Under review, IEEE *Access*.