Bachelor Computer Science & Datascience and Artificial Intelligence

Multi-Objective Optimisation Genetic Algorithm for Strategic Life Cycle Management Backlog Creation

Yvonne Dijksterhuis
s3537382

First supervisor and second supervisor:
Thomas Bäck & Camiel Koopmans

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

August 6, 2025

Abstract

The life cycle management (LCM) process of IT landscapes with a large variety in physical and virtual infrastructure is a challenging task, especially when there are underlying dependencies and constraints that need to be taken into account. The periodic hardware migration backlog can be defined as a multi objective subset selection problem on a graphical representation of the IT landscape. Genetic algorithms have not previously been applied to this particular field but are known to perform well on multi objective optimisation (MOO) problems in general, which motivates their usage for this research. Various genetic algorithms (GA) with Pareto rank based selection operators are constructed for the backlog creation. Experimental results demonstrate that all algorithms perform well on the smaller test landscape and are able to approximate the theoretical Pareto front, but the quality of the solutions vary on larger and more complex landscapes. Pareto based genetic algorithms are able to effectively generate a range of equally optimal solutions and can be applied as a tool to support decision making in short and long term LCM planning.

Contents

1 Introduction

SSC-ICT is one of the Dutch government entities responsible for overseeing the operations of the national data centres. One of its major operational programs is the Life Cycle Management (LCM) of existing custom applications. In the last few years, SSC-ICT has dedicated much time and effort to migrate applications by replacing out-of-support hardware and rebuilding the applications in a new and more secure environment. In recent periods, the speed of the LCM has increased significantly. While this is a positive development, the increased migration speed has highlighted the inherent limits of an undirected migration approach. The current application landscape contains various types of physical servers and virtual applications, all with underlying dependencies which makes the creation of a quarterly migration backlog and overall strategy a complex issue that cannot be solved by traditional means.

For the creation of a LCM backlog, there are multiple constraints to be taken into account, such as the maximal workload capacity, the resulting number of connections between applications of unequal security level, and the urgency of hardware replacement. The backlog creation can therefore be seen as a multi-objective optimisation (MOO) problem [4]. This study proposes the application of a genetic algorithm (GA) to solve the multivariate optimisation problem. A genetic algorithm is a type of evolutionary algorithm (EA) which is inspired by the biological process of evolution [5]. A genetic algorithm is an iterative algorithm, where possible solutions make up the population. From this population the best solutions are selected and modified to generate increasingly better offspring solutions. A genetic algorithm generally starts with the creation of a random population of solutions. The individuals within the population then go through the iterative process of fitness evaluation, mutation, crossover and selection to determine which solutions are deemed fit enough to progress to the next generation. The process iterates until a termination criterion is reached, upon which the optimal solution of the last generation is returned as the final solution of the algorithm. Research has shown that EAs are effective and robust methods for tackling MOO problems [19]. The goal of this research is therefore to construct a model representing SSC-ICT's complex application landscape and to develop a suitable GA for the automated LCM backlog creation.

In this paper, a mathematical framework will be constructed following the properties of SSC-ICTs application landscape, upon which the design of the algorithms will be based. Using this framework, a smaller theoretical application landscape will also be constructed to analyse the desired behaviour of the algorithms and serve as a benchmark for their performances. A larger graph based on the mathematical framework will also be randomly generated to compare the performance of the constructed algorithms on a more complex landscape.

The rest of this paper is structured as follows: section 2 discusses related works in the field. In section 3 a mathematical framework is defined to describe the application landscape and a similar test graph is constructed to determine an analytical Pareto Front. Operators of the proposed GAs for this subset selection problem are given in section 4. The results of the various GAs on the previously defined test graph and true data are given and compared in section 5. Section 6 discusses the results and future work. Section 7 concludes.

2 Related works

Many real world problems are complex and often need to take multiple conflicting objectives into account. These problems are hard to solve using conventional optimisation techniques that result in exact solutions [16]. In the past decades however, much progress has been made in the field of multi-objective evolutionary algorithms (MOEA), which in turn have become one of the primary methods for solving multi-objective optimisation problems [20]. Examples of their application range from continuous optimisation problems in mechanics to telecommunication and cloud computing [12, 16, 3].

At the time of writing, no literature on the application of MOEAs specifically on LCM backlog creation or LCM strategy in general can be found. However, in a similar sphere of the optimisation problem posed in this paper, Kumar et at. (2014) [6] successfully applied genetic algorithms to generate an Agile Release Planning with regards to multiple objectives and constraints. In their research however, only one of the objectives is taken into account for the fitness evaluation. This is similar to the weighted sum approach for general genetic algorithms, which reduces the dimension of a multi-objective problem, resulting in a scalar function as the fitness function that is to be evaluated. But in doing so, information about the relationship between the objective functions is lost. This method also introduces more parameters to the system that need to be tuned for the algorithm to perform as desired [15, 1]. For this research, it is preferable to be able to explain the optimality of a solution with regards to the properties of the objects within the application landscape. A popular alternative for the selection in multi-objective GAs are the Non-dominated Selection Genetic Algorithm (NSGA) algorithms [2].

In contrast to the weighted sum approach, NSGAs do not reduce the dimension of the problem for the evaluation of the problems, but rather determines the optimal solutions with regards to the trade-off between each of the optimisation objectives. The NSGA algorithm is introduced in 1994 by Srinivas and Deb [18] and is improved upon by Deb et. all [2] in 2002 with the Non-dominated Sorting Genetic Algorithm II. Both the NSGA and the NSGA-ii algorithm perform selection based on the Pareto principle of dominance, in which the optimality of solutions takes the values across all optimisation objectives into account. In NSGA algorithms, non-dominated solutions are seen as genetically fitter than those who are. The NSGA-ii improves on the computational complexity of the original NSGA algorithm and introduces both the concepts of crowding distance and elitism to the selection process. The crowding distance is calculated in addition to the Pareto ranking as a method to sort solutions that have the same ranking. This is important for the elitist selection, which is done by utilising $(\mu + \lambda)$-selections where both the μ parents and the λ offsprings are taken into account for the selection. The $(\mu + \lambda)$-selection is effective as it ensures non-deterioration of solutions fitness within the population. In the recent decades, the NSGA based algorithms have become a popular research method when dealing with multi-objective optimisation problems. The widespread application of NSGA based algorithms and their demonstrated effectiveness across research applications [9] motivates the use of a NSGA based algorithm for this research. In particular, this study utilises a hybrid variation between the original NSGA and the NSGA-ii algorithms, with a naive ranking algorithm as in the original NSGA, but with the crowding distance and elitism of the NSGA-ii algorithm to ensure non-deterioration of the population.

3 Application Landscape Model

In the model of the application landscape, two types of objects are discerned: an *Application* objects (AO) and *Management* objects (MO). An AO consists of all the hardware and virtual components that make up an application environment, these are components such as physical (database) servers and network cables but also software packages and virtual machines. The final application for users consists of one or more AOs.

A Management object consists of the physical and virtual components that make up a management environment from which applications can be monitored and managed. Each MO manages one or more AOs, which are not necessarily part of the same application.

In addition to different object types, the environments in which the objects lie can also differ. For this model, two environments are defined with unequal security levels: an *old* environment with a lower security level and a *new* environment with a higher security level. The goal of the LCM operation is to eventually replace all AOs and MOs in the old environment with new AOs and MOs in the new environment. It is possible for two objects in different environments to be connected. However, network connections between the two environments are undesirable as they create points of weakness for the security of the new environment and should therefore be minimised or removed if possible.

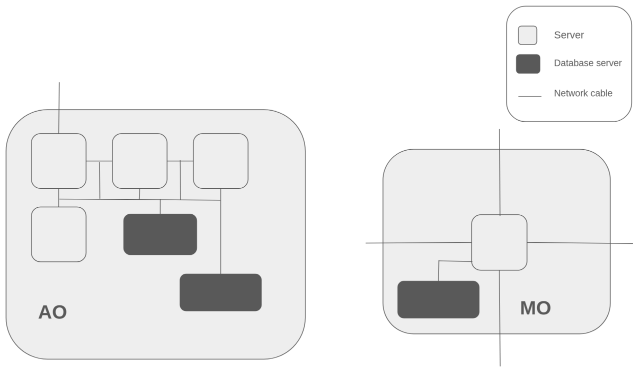


Figure 1: Schematic overview of possible application and management object configurations.

3.1 Graphical Representation of the Application Landscape

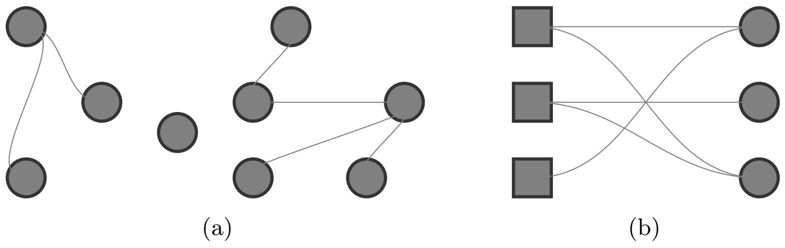


Figure 2: Examples of disjoint (a) and bipartite (b) graphs.

To understand the graphical representation of the application landscape, a few mathematical definitions related to graphs are given. A *graph* is defined as a pair $G = (V, E)$ of sets where V is a set of vertices and E is a set of unordered pairs $\{v_1, v_2\} \in V$ of all edges, or links, between two vertices. The *cardinality* of a set V is the number of elements in the set and it is noted as $|V|$. A graph is called *bipartite* if there exists two subsets A and B of V such that each intersection is empty, $A \cap B = \emptyset$, their union equals V, $A \cup B = V$, and each edge connects a vertex in A to a vertex in B. Figure 2b shows a bipartite graph with two disjoint sets of vertices denoted by squares and circles.

A *subgraph* $G' = (V', E')$ of G is written as $G' \subseteq G$ with $V' \subseteq V$ and $E' \subseteq E$, meaning that G' is *contained* in G. Lastly, two subgraphs $G' = (V', E')$ and $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ of G are called *disjoint subgraphs* of G if that $V' \cap \widetilde{V} = \emptyset$ and $E' \cap \widetilde{E} = \emptyset$. In figure 2a an example graph consisting of three disjoint subgraphs is shown.

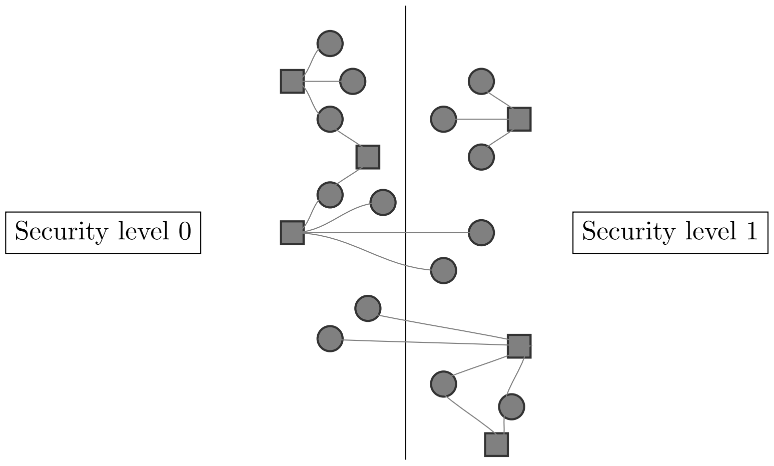


Figure 3: A possible representation of the application landscape where the vertical line marks the border between two environments of varying security levels.

For this research, the application landscape is represented as a bipartite graph $G = (V, E)$, where V, the set of vertices, is defined as the disjoint union of $V = A \uplus M$. Here, M is defined as the set of abstract *Management* objects and A the set of abstract *Application* objects. The set E contains the edges from $m \in M$ to one or more $a \in A$, indicating that m manages a. An example of this representation is given in Figure 3, where the management objects are represented as square vertices and the application objects are represented as round vertices.

3.2 Optimisation Objectives

A bipartite graph $G = (V, E)$ as described in section 3.1 is defined for the theoretical representation of the application landscape. G has disjoint subsets of vertices M and A representing the sets of management and application objects respectively. There are three main requirements for the LCM backlog creation. The first requirement is to increase security by migrating the oldest hardware components, the second is to mitigate existing risks by removing cross level connections, and the last requirement is to select a set of objects that can be migrated realistically within the given time frame. These constraints will form the basics for the mathematical model.

Let C denote the maximal migration capacity per quarter. Integer values $u, w \in \mathbb{N}_{\geq 0}$ are assigned to each $v \in V$ where u denotes the urgency of migration, w the estimated workload. A variable $r \in \{0, 1\}$ is assigned to each $v \in V$ which specifies the security level of the represented object. The value r is assigned by the function

$$r_v = \begin{cases} 0 & \text{if } v \text{ lies in the new environment} \\ 1 & \text{otherwise} \end{cases}$$

With these variables, a subset selection problem can be defined on G where at each time period a finite subset S of V with $|S| = k \in \mathbb{N}_{>0}$ is selected such that the total urgency of the selected set

$$U(S) := \sum_{s \in S} u_s$$

is maximised with the constraint that the total estimated workload

$$W(S) := \sum_{s \in S} w_s$$

satisfies the constraint

$$W(S) \leq C.$$

The function $R(S)$ for the overall risk-level is defined as follows:

$$R(S) := \sum_{(a,m)\in E} \lfloor \frac{r_a + r_m}{2} \rfloor \cdot |\mathbb{1}_S(a) - \mathbb{1}_S(m)| + |r_a - r_m| \cdot \left\lceil \frac{|r_a - \mathbb{1}_S(a)| + |r_m - \mathbb{1}_S(m)|}{2} \right\rceil$$

The function $R(S)$ iterates over all the edges within the graph G, and counts the number of cross-environment edges after a solution S is applied. For any starting configuration of the graph G and any subset $S \subset V$, there are only two cases which results in cross-environment connections: The first case is when both the application object and the management object start in the lower security level, and only one of the two is chosen to be migrated. Since it is not possible to migrate an object to a lower security level, the case where both objects already lie in the new environment with the higher security level is irrelevant. The second case is when only one of the two objects lie in the new environment and the object which lies in the old environment is not chosen as part of the solution. In this latter case, an existing cross-environment connection is perpetuated while a new cross-environment connection is created in the former. These two cases are mutually exclusive, and this exclusivity is represented by the operations

$$\lfloor \frac{r_a + r_m}{2} \rfloor \text{ and } |r_a - r_m|$$

within the summation. The first operation results in a value of 1 if and only if both objects lie in the old environment and zero otherwise while the second results in a value of 1 if and only if the two objects start out in different environments and zero otherwise. For the final part of the function which corresponds to the second case

$$\left\lceil \frac{|r_a - \mathbb{1}_S(a)| + |r_m - \mathbb{1}_S(m)|}{2} \right\rceil$$

it is again important to remember that an object that lies in the new environment cannot be chosen as part of the solution S. Therefore the nominator of the fraction always has a maximum value of 1.

3.2.1 Formal Problem Definition

Finally, with the functions defined above, a minimisation model can be defined as follows:

$$\min_{S\subseteq V}(R(S), U(S))$$

Subject to

$$C \geq W(S) \geq 1$$

This minimisation model does not necessarily have an unique optimal solution. Each possible solution represents a trade-off between conflicting objectives within the model, which introduces the concept of *dominance* for solutions within a MOO model.

Definition 3.1

For a minimisation problem with N objectives, a solution s_i is said to be *dominant* over another solution s_j if:

1. solution s_i is equal to or better than s_j for all N objective functions: $f_n(x_i) \leq f_n(s_j)\ \forall n = 1, 2, \ldots, N$.
2. solution s_i is strictly better than s_j for at least one objective function: $\exists n \in \{1, \ldots, N\}$ such that $f_n(s_i) < f_n(s_j)$.

A solution is called *Pareto optimal* if there does not exist an alternative solution which minimises one component without compromising the minimality of another component. Dominant solutions are Pareto optimal for at least one variable. The set containing all dominant solutions is a Pareto optimal set and can be used to represent the Pareto front. Figure 4 illustrates the concept of Pareto dominance between solutions of a two dimensional minimisation problem.

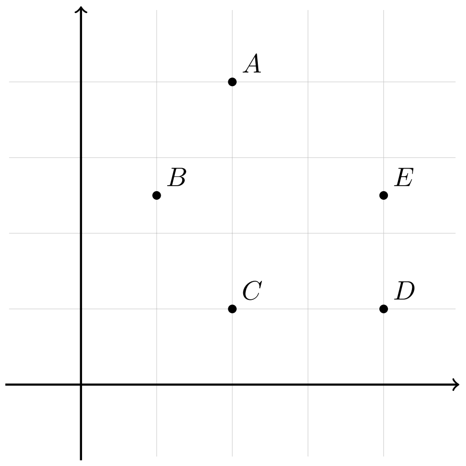


Figure 4: An example of dominance in a minimisation problem. Point C and B are non-dominated points. A and D are dominated by C, while E is dominated by both B and D.

3.3 Test Graph and the Analytical Pareto Front

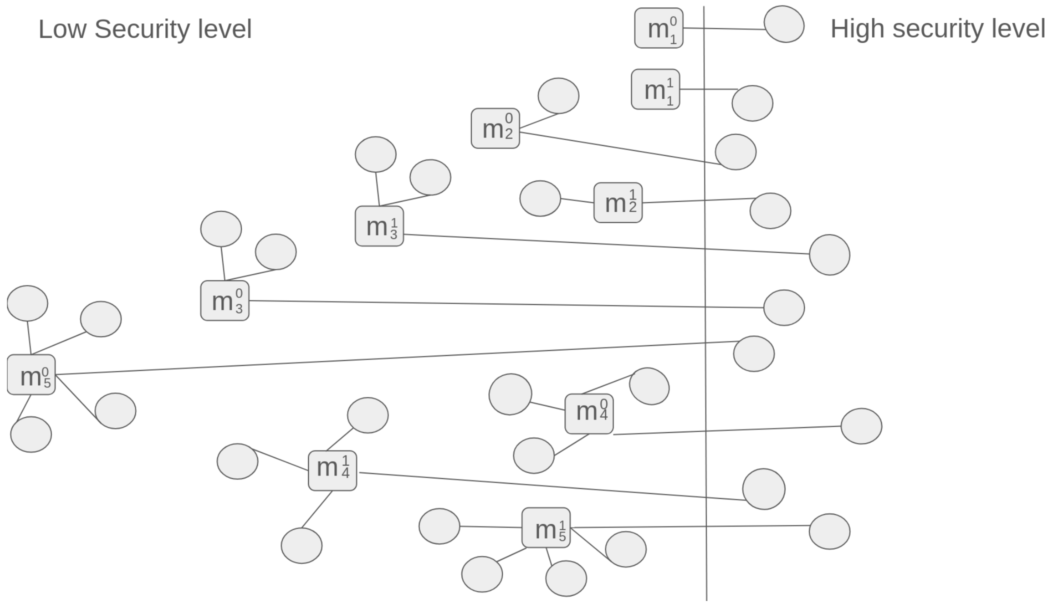


Figure 5: Parts of the test graph G_T showing the clusters of m_i^k with their connected application objects a_{ij}^k for $k = 0, 1$.

In this section, a theoretical landscape is constructed with desirable properties to allow for analytical analysis on the structure of the exact Pareto front. The Pareto front of this graph can be separated into two parts: an analytical and non-analytical part. The analytical part of the front will be shown to follow a fixed structure, while the non-analytical part of the Pareto front is found manually, as those do not necessarily follow a fix pattern that can be easily proven. Arguments will be given on the relation in terms of Pareto dominance between the solutions of the analytical front and solutions with properties similar to those of the non-analytical front. Exact proofs for the structure of non-analytical solutions is considered out of scope for this study.

Let G_T be a bipartite graph similar to the graph defined in 3.1, with the exception that $u, w \in \mathbb{Q}$ rather than $\mathbb{Z}$. Let $V = A \cup M$, where

$$M = \{m_i^k | k = \{0, 1, ..., 4\}, i \in \{1, 2, ..., 5\}\} \text{ and } A = \{a_{ij}^k | k = \{0, 1, ..., 4\}, i, j \in \{1, 2, ..., 5\}\}.$$

Furthermore, let the set of edges E be defined as

$$E = \{(m_i^k, a_{ij}^k) |\ m_i^k \in M, a_{ij}^k \in A\}.$$

For each of node in G_T, the values for the workload, urgency and risk are defined as follows: Firstly, the location value $r_v = 0$ is assigned to all application nodes $a_{i1}^k \in A$, such that each management object has one connected application object in another security level. All other nodes are assigned the location value $r_v = 1$. Secondly, each management object $m_i^k \in M$ and its connected application objects a_{ij}^k share the same workload $w_{m_i^k} = w_{a_{ij}^k} = 2^k$. Lastly, for the urgency value, all application objects $a_{ij}^k \in A$ are assigned urgency $u_{a_{ij}^k} = 0$, while the urgency for the management objects are assigned following the function:

$$u_{m_i^k} = \begin{cases} 2 - \dfrac{1}{i+1} & \text{if } k = 0, \\ 3 \cdot 2^k - 2 + \left(1 - \dfrac{1}{\frac{k}{5} + i + 1}\right) & \text{if } k > 0. \end{cases}$$

Figure 5 illustrates a part of the graph G_T and Table 1 gives an overview of the urgency values associated with the management objects $m_{ij}^k \in M$.

Table 1: Values for the urgencies $u_{m_i^k}$ of the graph G_T for $i = 1, \dots 5$ and $k = 0, \dots 4$, where each of the m_i^k has a workload of 2^k.

$i \backslash k$	0	1	2	3	4
1	$1\frac{1}{2}$	$4\frac{6}{11}$	$10\frac{7}{12}$	$22\frac{8}{13}$	$46\frac{9}{14}$
2	$1\frac{2}{3}$	$4\frac{11}{16}$	$10\frac{12}{17}$	$22\frac{13}{18}$	$46\frac{14}{19}$
3	$1\frac{3}{4}$	$4\frac{16}{21}$	$10\frac{17}{22}$	$22\frac{18}{23}$	$46\frac{19}{24}$
4	$1\frac{4}{5}$	$4\frac{21}{26}$	$10\frac{22}{27}$	$22\frac{22}{28}$	$46\frac{23}{29}$
5	$1\frac{5}{6}$	$4\frac{26}{31}$	$10\frac{27}{32}$	$22\frac{28}{33}$	$46\frac{29}{34}$

To analyse potential Pareto fronts on G_T, a minimisation problem with constraints needs to be defined first. Let the maximal capacity C be set to 31, this results in the minimisation problem P:

$$\min_{S \subseteq V} (R(S), U(S))$$
$$\text{Subject to } 31 \geq W(S) \geq 1$$

The solutions on the Pareto front of P can be determined by the properties of G_T. To this end, two propositions are given.

Proposition 3.2

Let $S \subset M$ be a set of management objects with maximal workload $W(S) \leq 2^k$ then it holds for the urgency of S that

$$U(S) \begin{cases} < 3 \cdot 2^k - 2 & \text{if } \nexists i \in \{1, 2, \dots, 5\} \text{ such that } m_i^k \in S \\ \geq 3 \cdot 2^k - 2 & \text{otherwise} \end{cases}$$

Proof. The proof will be given by means of induction. Given a set $S \subset M$ with $W(S) =\ 2^k$, consider the base cases where $k = 0$.

For $k = 0$, there are two possible solutions. Due to the workload constraint, S is either the empty set with $U(\emptyset) = 0 < 1 = 3 \cdot 2^0 - 2$ or a singleton set $S = \{m_i^0\}$ with workload $W(S) = 2^0$ and urgency

$$U(S) = 2 - \frac{1}{i+1} > 1 = 3 \cdot 2^0 - 2, \forall i \in \mathbb{N}.$$

So the statement holds for the base cases. Now assume that the statement holds for all $k = 0, 1, ..., K-1$. This serves as the induction hypothesis. Regard the case where $k = K$. Then S contains at most 1 object m_i^K with workload 2^K. If it does, then S is a singleton $S = \{m_i^K\}$ with urgency

$$U(S) = 3 \cdot 2^K - 2 + (1 - \frac{1}{\frac{K}{2} + i + 1}) > 3 \cdot 2^K - 2$$

If S is not the singleton set of an object with weight 2^k, then S can be split into two subsets S_1, and S_2, each with a maximal workload with workload of 2^{K-1} [10]. So $W(S) \leq 2^K$, and by the induction hypothesis:

$$U(S) \leq U(S_1) + U(S_2) \overset{ind.hyp}{<} 2(3 \cdot 2^{K-1} - 2) < 3 \cdot 2^K - 2$$

The inequality also holds for both possible cases where $k = K$, so it will hold for all k. □

An important result that follows from proposition 3.2 is that all solutions S that do not contain an element of the workload class k and have a workload $W(S) \leq 2^k$ will be dominated by the singleton solution $S' = \{m_1^k\}$. Furthermore, with the above proposition, claims can be made about the non-dominated solutions of P.

Proposition 3.3

Let K be the highest workload class. Given the workload constraint $C = 2^{K+1} - 1$, any solution S' that contains only management objects but does not contain a management object of each workload class $k = 0, 1, \ldots, K$ is dominated by a solution S that does contain management objects from each workload class $k = 0, 1, \ldots, K$.

Proof. Let $S' \subset M$ be a solution that does not contain a management object of each workload class $k = 0, 1, \ldots, K$. Consider the following construction of a dominant solution S from S' which does contain a management object from each workload class: starting from $k = K - 1$, if S' contains more than one management objects in this class, then replace two objects m_i^{K-1}, m_j^{K-1} with one management object m_1^K or m_2^K. Repeat this process until there is at most one management object of the workload class $k = K - 1$, then move on to the workload class $k = K - 2$ to repeat this process until $k = 0$ is reached. If there is a workload class k for which S does not have a management object at the end of this process, then add the object m_1^k. This addition simultaneously increases the total urgency of S and decreases the risk value. The addition does not violate the workload constraint, as the constraint allows for maximally one object per workload class. This process ensures that S has at one management object per workload class at the end.

From proposition 3.2, it follows that the urgency of any two management object of workload class k is strictly lower than the urgency of the object, and by replacing two management objects from a lower workload class with either m_k^1 or m_k^2, the total risk of the solution either decreases or remains unchanged. From these two points, it is possible to conclude that S has a strictly higher urgency value than S' for the same risk value or less. Thus S' is dominated by S. □

Proposition 3.3 gives a restriction on the properties of non-dominated solutions, but it does not give the exact form of the solutions that lie on the Pareto front. Let $S_{P_{\max}}$ and $S_{P_{\min}}$ be two solutions that lie on the extremes of the analytical part of the Pareto front for P on G_T.

These two points form the upper and lower boundaries for the other solutions on the analytical Pareto front. In combination with the above propositions, two other conditions can be set for the optimal solutions $S_{P_{\min}}$ and $S_{P_{\max}}$:

1. The solution $S_{P_{\min}}$ is the non-dominated solution with the highest possible urgency, with the constraint that $R(S_{P_{\min}})$ is minimal on the analytical front;
2. The solution $S_{P_{\max}}$ is the non-dominated solution with the highest urgency, without any constraints on $R(S_{P_{\max}})$.

Corresponding to the constraints for $S_{P_{\min}}$ and propositions 3.2 and 3.3, the optimal approach to construct the solution would be to take as many of the complete clusters of m_i^k and a_{ij}^k as the maximal capacity allows. By construction of the graph G_T, the application objects connected to the management objects $m_1^k, k = 0, 1, \ldots, 5$ already lie in the new environment. Thus the only solution that contains an object from each workload class $k = 0, 1, \ldots, 5$ and does not introduce new risks into the system is the solution containing all the management objects with only one connected application object:

$$S_{P_{\min}} = \{m_1^k | k \in [0, 4]\}$$

With the corresponding values

$$\begin{aligned}
W(S_{P_{\min}}) &= 1 + 2 + 4 + 8 + 16 && = 31\\
U(S_{P_{\min}}) &= 1\frac{1}{2} + 4\frac{6}{11} + 10\frac{7}{12} + 22\frac{8}{13} + 46\frac{9}{14} && = 85\frac{10655}{12012}\\
R(S_{P_{\min}}) &= 25 - 5 && = 20
\end{aligned}$$

Note that all solutions will need to take both the existing and potentially new cross-environment connections into account for the risk value. Similarly, an optimal strategy to determine $S_{P_{\max}}$ is to take as many of the management objects with the highest urgencies as the maximal capacity allows from the column with the maximal k value. Then, if the maximal capacity is not yet reached, to move on to the management object with the highest urgency from the column with the next highest k value and so on. By proposition 3.3, this strategy ensures that $S_{P_{\max}}$ is the set with the highest possible total urgency within the constraints of the maximal capacity. For G_T with $C = 31$, this strategy results in the set

$$S_{P_{\max}} = \{m_5^k | k \in [0, 4]\}$$

with

$$\begin{aligned}
W(S_{P_{\max}}) &= 1 + 2 + 4 + 8 + 16 && = 31\\
U(S_{P_{\max}}) &= 1\frac{5}{6} + 4\frac{26}{31} + 10\frac{27}{32} + 22\frac{28}{33} + 46\frac{29}{34} && = 87\frac{40295}{185504}\\
R(S_{P_{\max}}) &= 4 + 4 + 4 + 4 + 4 + 20 && = 40
\end{aligned}$$

Now that $S_{P_{\max}}$ and $S_{P_{\min}}$ are determined, the constraints on the rest of the solutions on the analytical Pareto front are known: Each solution S_{P_n} on the analytical front must have at least an urgency $U(S_{P_n}) > 85\frac{10655}{12012} \approx 85,887$ and a risk value $20 < R(S_{P_n}) \leq 40$. So starting from the solution $S_{P_{\min}}$, the solutions within the boundaries set by $S_{P_{\min}}$ and $S_{P_{\max}}$ can be constructed by increasing the risk value and finding the solution with the highest urgency level among the solutions with the same risk values.

Let S_{P_n} be a solution with risk $R(S_{P_n}) \leq n$ and $U(S_{P_n}) = \max_{S \subset V}\{U(S) | R(S) = n\}$. From the definition of S_{P_n}, it follows that $S_{P_0} =_{P\,\min}$ and $S_{P_{40}} = S_{P_{\max}}$. To find the other solutions S_{P_n}, two propositions are needed that define additional restrictions on their properties.

Proposition 3.4

If S_{P_n} is non-dominated and contains an element of each weight class k, then there are no two elements $m_i^k, m_j^\ell \in S_{P_n}$ where $k > \ell$ and $i > \ell$.

Proof. Let S_{P_n} be a non-dominated solution that does contain two elements $m_i^k, m_j^\ell \in S_{P_n}$ where $k > \ell$ and $i > j$. Now, let S'_{P_n} be equal to S_{P_n} with the exception that the two elements $m_i^k, m_j^\ell \in S_{P_n}$ are replaced with m_i^ℓ, m_j^k. Note that S'_{P_n} still has the same risk as S_{P_n}, but the urgencies are different. Consider the difference between the urgencies:

$$\begin{aligned}
U(S_{P_n}) - U(S'_{P_n}) &= U(\{m_i^k, m_j^\ell\}) - U(\{m_j^k, m_i^\ell\}) \\
&= u_{m_i^k} + u_{m_j^\ell} - u_{m_j^k} - u_{m_i^\ell} \\
&= \left(1 - \frac{1}{\frac{k}{5} + i + 1}\right) + \left(1 - \frac{1}{\frac{\ell}{5} + j + 1}\right) - \left(1 - \frac{1}{\frac{k}{5} + j + 1}\right) - \left(1 - \frac{1}{\frac{\ell}{5} + i + 1}\right) \\
&= \frac{25(j-i)(k-l)(10 + 5i + 5j + k + l)}{(5 + 5i + k)(5 + 5j + k)(5 + 5i + l)(5 + 5j + l)} \\
&< 0
\end{aligned}$$

The difference is negative, since $i > j$, meaning that S_{P_n} is dominated by S'_{P_n}. This contradicts the assumption that S_{P_n} is non-dominated. Thus any non-dominated solution S_{P_n} cannot contain elements pairs of elements $m_i^k, m_j^\ell \in S_{P_n}$ where $k > \ell$ and $i > \ell$. ☐

The second proposition introduces further restrictions on the elements m_i^k and an element of its direct subsequent workload class m_j^{k+1}.

Proposition 3.5

If S_{P_n} is non-dominated and contains an element of each weight class k, then it does not contain elements m_i^k and m_j^{k+1} where $i \geq j + 2$.

Proof. Let S_{P_n} be a non-dominated solution that does contain two elements $m_i^k, m_j^{k+1} \in S_{P_n}$ where $i \geq j + 2$. Then i can be written as $i = j + a$, $a \in \mathbb{N}_{\geq 2}$. Now, let S'_{P_n} be equal to S_{P_n} with the exception that the two elements $m_i^k, m_j^{k+1} \in S_{P_n}$ are replaced with m_{i-1}^k, m_{j+1}^{k+1}. Consider the difference between the urgencies:

$$\begin{aligned}
U(S_{P_n}) - U(S'_{P_n}) &= U(\{m_{j+a}^k, m_j^{k+1}\}) - U(\{m_{i-1}^k, m_{j+1}^{k+1}\}) \\
&= u_{m_{j+a}^k} + u_{m_j^{k+1}} - u_{m_{j+a-1}^k} - u_{m_{j+1}^{k+1}} \\
&= \left(1 - \frac{1}{\frac{k}{5} + j + a + 1}\right) + \left(1 - \frac{1}{\frac{k+1}{5} + j + 1}\right) - \left(1 - \frac{1}{\frac{k}{5} + j + a}\right) - \left(1 - \frac{1}{\frac{k+1}{5} + j + 2}\right) \\
&= \frac{25(6 - 5a)(5a + 10j + 2k + 11)}{(5j + k + 6)(5j + k + 11)(5a + 5j + k)(5a + 5j + k + 5)} \\
&< 0
\end{aligned}$$

The difference is strictly negative, since $a \geq 2$, meaning that S_{P_n} is dominated by S'_{P_n}. This contradicts the assumption that S_{P_n} is non-dominated. Thus any non-dominated solution S_{P_n} cannot contain elements pairs of elements elements m_i^k and m_j^{k+1} where $i \geq j + 2$. ☐

Lastly, note that increasing the risk value by 1 will always yield a higher urgency value than the possible values for solutions risk value n. Thus S_{P_n} is the solution with the highest urgency for risk value n.

Let $S_{P_{20}} = S_{P_{\min}}$. Using the properties of optimal solutions as determined by proposition 3.4 and 3.5, the S_{P_n} can be constructed by steadily increasing the risk value n and finding the solution with the maximal urgency value of all solutions with risk $20 < n \leq 40$ as follows: Let I be the value corresponding to the maximal number of application objects connected to an management object. Each possible risk value n can be uniquely written as

$$n = I \cdot b + c + 20 \text{ with } b, c \in \mathbb{N} \text{ and } 0 \leq b \leq K.$$

To construct S_{P_n}, take the elements m^k_{b+2}, where $k < c$. And for each $k \geq c$, take the elements m^k_{b+1}. This construction ensures that the restrictions for non-dominated solutions set by proposition 3.4 and 3.5 are satisfied. Furthermore, this method selects one element from each workload class, corresponding to the properties of non-dominated solutions as shown by proposition 3.3. And this process always selects the object with the highest possible urgency value given the risk constraint.

Table 2 shows two solutions on the Pareto front, using urgency values to the represent the corresponding management objects. To illustrate the construction of optimal solutions, look at the example construction of $S_{P_{\max}}$ of the test graph G_T with $I = 5$:

$$S_{P_{40}} = S_{P_{\max}} = \{m^0_5, m^1_5, m^2_5, m^3_5, m^4_5\}.$$

The risk value 40 can be represented as $40 = 5 \cdot 4 + 0 + 20$ Then, for all $k = 0, 1, \ldots, 4$ the elements that should be selected for the solution S are all the m^k_{4+1}, resulting in

$$S = \{m^k_5 | \forall k = 0, 1, \ldots, 4\} = S_{P_{\max}}$$

Table 2: Urgency values of the management objects representing examples of non-dominated solution for the test graph G_T. From top to bottom, the solutions $S_{P_{28}}$ and $S_{P_{\max}}$ are coloured.

$i \backslash k$	0	1	2	3	4
1	$1\frac{1}{2}$	$4\frac{6}{11}$	$10\frac{7}{12}$	$22\frac{8}{13}$	$46\frac{9}{14}$
2	$1\frac{2}{3}$	$4\frac{11}{16}$	$10\frac{12}{17}$	$22\frac{13}{18}$	$46\frac{14}{19}$
3	$1\frac{3}{4}$	$4\frac{16}{21}$	$10\frac{17}{22}$	$22\frac{18}{23}$	$46\frac{19}{24}$
4	$1\frac{4}{5}$	$4\frac{21}{26}$	$10\frac{22}{27}$	$22\frac{23}{28}$	$46\frac{24}{29}$
5	$1\frac{5}{6}$	$4\frac{26}{31}$	$10\frac{27}{32}$	$22\frac{28}{33}$	$46\frac{29}{34}$

The solutions that follow the above structure are assumed to be Pareto optimal. It is, however, difficult to prove that the S_{P_n} are not dominated by solutions of all other possible compositions. Thus, as opposed to a proposition, a conjecture is given.

Conjecture 3.6

Let K be the highest workload class. Given the workload constraint $C = 2^{K+1} - 1$, the solutions containing one management object from each workload class $k = 0, 1, \ldots, K$ are not dominated by any solution S' with equal risk that contains a application objects.

A few arguments in support of this conjecture are given. First, consider that application objects have workloads but no urgency. Starting from a S solution with one management object from each workload class and a risk value n, a management object with a workload value k can be traded for two objects with workload $k - 1$ (or more for lower workload classes) to construct S'. From the results of proposition 3.2, it follows that the sum any two management objects from a lower workload class $k - 1$ will always have a lower urgency level than that of m^k_1. Following this logic, solutions containing application objects will have even lower urgency values for equal or higher risk values as the original solution, depending on how many connected objects are taken along to counter the increase in risk.

It is particularly the case where S' contains an element m^K_i and S contains an element m^K_j with $j \leq i$ that is of interest. It is suspected that the slight advantage in urgency that S' would have from the object of workload K with a higher urgency than that of S, does not outweigh difference in urgency of objects from the lower workload classes. Especially since S' cannot have a management object from each workload class due to the workload constraint and, as opposed to S, it contains one or more application objects without urgency.

Furthermore, if there exists a counterexample for the conjecture, it is possible to adjust the test graph slightly to ensure its compliance. This can be done by assigning a much higher workload value to each application object in comparison to the management objects. Doing so, the addition of any application object will severely impact the number of objects with non-zero urgency that can be included in the solutions.

Lastly, to finish the construction of the exact Pareto front, the solutions of the non-analytical graph are solutions with risk values lower than the risk value of $S_{P_{\min}}$. There are three of these solutions in total for the test graph G_T with risk values 17, 18 and 19. They are found manually and the exact composition of these solutions are shown in Table 3, 4 and 5.

Table 3: Urgency values of the management objects representing the optimal solution with risk 17. Management objects in the solution are shown in bold. Coloured boxes mean that all the application object of this management object are also in the solution.

$i\backslash k$	0	1	2	3	4
1	$\mathbf{1\frac{1}{2}}$	$\mathbf{4\frac{6}{11}}$	$\mathbf{10\frac{7}{12}}$	$\mathbf{22\frac{8}{13}}$	$46\frac{9}{14}$
2	$\mathbf{1\frac{2}{3}}$	$\mathbf{4\frac{11}{16}}$	$10\frac{12}{17}$	$22\frac{13}{18}$	$46\frac{14}{19}$
3	$\mathbf{1\frac{3}{4}}$	$4\frac{16}{21}$	$10\frac{17}{22}$	$22\frac{18}{23}$	$46\frac{19}{24}$
4	$\mathbf{1\frac{4}{5}}$	$4\frac{21}{26}$	$10\frac{22}{27}$	$22\frac{23}{28}$	$46\frac{24}{29}$
5	$\mathbf{1\frac{5}{6}}$	$4\frac{26}{31}$	$10\frac{27}{32}$	$22\frac{28}{33}$	$46\frac{29}{34}$

Table 4: Urgency values of the management objects representing the optimal solution with risk 18. Management objects in the solution are shown in bold. Coloured boxes mean that all the application object of this management object are also in the solution.

$i\backslash k$	0	1	2	3	4
1	$1\frac{1}{2}$	$\mathbf{4\frac{6}{11}}$	$\mathbf{10\frac{7}{12}}$	$\mathbf{22\frac{8}{13}}$	$46\frac{9}{14}$
2	$\mathbf{1\frac{2}{3}}$	$4\frac{11}{16}$	$\mathbf{10\frac{12}{17}}$	$22\frac{13}{18}$	$46\frac{14}{19}$
3	$\mathbf{1\frac{3}{4}}$	$4\frac{16}{21}$	$10\frac{17}{22}$	$22\frac{18}{23}$	$46\frac{19}{24}$
4	$\mathbf{1\frac{4}{5}}$	$4\frac{21}{26}$	$10\frac{22}{27}$	$22\frac{23}{28}$	$46\frac{24}{29}$
5	$1\frac{5}{6}$	$4\frac{26}{31}$	$10\frac{27}{32}$	$22\frac{28}{33}$	$46\frac{29}{34}$

Table 5: Urgency values of the management objects representing the optimal solution with risk 19. Management objects in the solution are shown in bold. Coloured boxes mean that all the application object of this management object are also in the solution.

$i\backslash k$	0	1	2	3	4
1	$\mathbf{1\frac{1}{2}}$	$\mathbf{4\frac{6}{11}}$	$\mathbf{10\frac{7}{12}}$	$22\frac{8}{13}$	$\mathbf{46\frac{9}{14}}$
2	$\mathbf{1\frac{2}{3}}$	$\mathbf{4\frac{11}{16}}$	$10\frac{12}{17}$	$22\frac{13}{18}$	$46\frac{14}{19}$
3	$\mathbf{1\frac{3}{4}}$	$4\frac{16}{21}$	$10\frac{17}{22}$	$22\frac{18}{23}$	$46\frac{19}{24}$
4	$1\frac{4}{5}$	$4\frac{21}{26}$	$10\frac{22}{27}$	$22\frac{23}{28}$	$46\frac{24}{29}$
5	$1\frac{5}{6}$	$4\frac{26}{31}$	$10\frac{27}{32}$	$22\frac{28}{33}$	$46\frac{29}{34}$

4 Proposed Genetic Algorithms

The GAs studied in this research all use the same Pareto rank based selection operator while the other evolutionary operators vary. For the mutation operator, bit flip mutation and inversion mutation are implemented and for the crossover operator, one-point and two-point crossovers are implemented. Figure 6 illustrates the steps within a genetic algorithm.

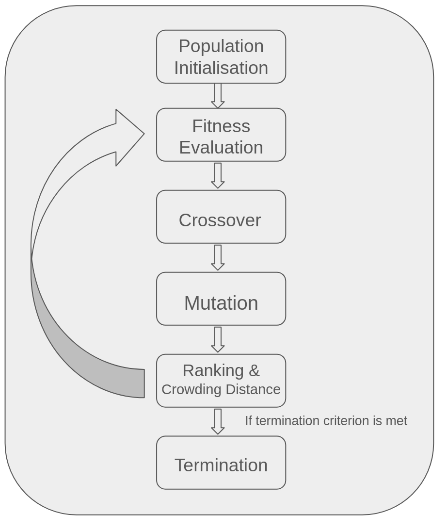

Population Initialisation
Fitness Evaluation
Crossover
Mutation
Ranking & Crowding Distance
If termination criterion is met
Termination

Figure 6: Schematic overview of the steps in the proposed genetic algorithms.

4.1 Chromosome Representation

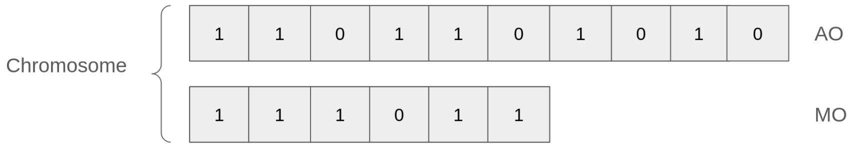


Figure 7: Example of a chromosome with binary representations of the selected application and management objects.

In GAs, a *chromosome* represents a possible solution within the search space. For the specified subset selection problem a binary encoding is chosen for the representation of the chromosomes. A class object is defined with two boolean arrays as member variables, one representing the set of AOs and one representing the set of MOs. This is illustrated in Figure 7. Each position a_i in an array A corresponds to an object within the corresponding set of objects. For each position a_i within an array, the value assignment is as follows:

$$a_i = \begin{cases} 1 & \text{if the object is part of the solution} \\ 0 & \text{otherwise} \end{cases}$$

4.2 Initialisation

The initialisation of the first population is important for the performance of a GA. If the initial population contains too many solutions that do not satisfy the constraints of the minimisation problem, the likelihood of creating valid offsprings is impacted and the GA could potentially fail to reach a satisfying result in reasonable time. This is especially probable when the search space is large. Such situations can be avoided by setting the probability p_{init} based on the given capacity constraints. At initialisation, a chromosome population of size μ is randomly generated by iterating over an array containing the IDs of all objects within the application landscape, and selecting each one with probability p_{init}. With this process, the number of selected objects in each chromosome is binomially distributed. So the expected number of selected objects per chromosome is $N \cdot p_{init}$ with N the total number of applications within the landscape. To ensure that the initial population of chromosomes contains a reasonable number of randomly selected objects, the probability p_{init} can be set according to the equation:

$$p_{init} = \frac{\alpha C_{\max}}{N},$$

where $C_{\max}$ is the maximal capacity and $\alpha \in (0, 1)$. Then the expected number of objects within a chromosome is equal to αC. It is preferable to have a lower expected number of objects than the maximal capacity, because the true number of objects selected within a chromosome will vary around the expected value. So setting $\alpha C_{\max} < C_{\max}$ as the expected value could increase the probability of generating valid initial solutions. An obvious choice for α would be to set it to a value such that $\alpha C_{\max}$ lies between the upper bound $C_{\max}$ and the lower bound $C_{\min}$, of the optimisation constraint. However, it is important to note that this value of α uses an implicit assumption that each object has a workload of 1, which is not (always) the case. If the average workload of all objects within the graph are known, then an improvement would be to adjust the value of $\alpha C_{\max}$ to incorporate the average workload of the objects.

For example, let w_{avg} be the average workload of all objects within a set V of nodes in G, and let $C_{\max}$ and C_{min} be the upper and lower bound of the optimisation constraint defined on G. A suitable choice for the initial probability, with regards to the average workload of objects within V is then given by

$$p_{init} = \frac{C_{\max} + C_{\min}}{2N \cdot w_{avg}}$$

The above function improves the probability of generating valid solutions for the initial population, but the initialisation remains a stochastic process and it is possible that the workloads of the solutions still do not satisfy the constraint. A trimming function can therefore be added as a last step to the initialisation process to ensure that the total workload of each solution remains below the maximal capacity constraint.

4.2.1 Initial Solution Trimming

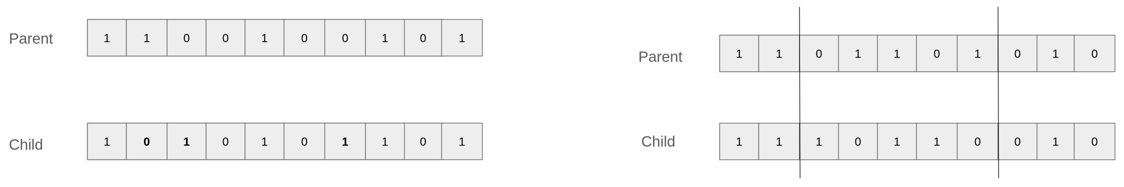
The trimming function for the initialisation works as its name implies: if the workload of a chromosome is larger than a specified value $\alpha C_{\max}$ then one or more objects are randomly removed from the solution. This process repeats until the workload of a solution is less than or equal to $\alpha C_{\max}$. In the context of this optimisation problem, the upper capacity limit is a harsher constraint than the lower limit, since it is more acceptable for a team to perform slightly less work than it is to ask them to perform more work than they are capable of. Thus, to avoid additional complexity in the initialisation process the choice is made to not adjust the solutions that have a total workload that is lower than the minimal capacity C_{min}. The fitness of these solutions, and other offsprings that do not satisfy the optimisation constraints are adjusted with the help of the penalty function defined in section 4.6.1.

4.3 Mutation

The mutation operator is a probabilistic operator that changes small parts of a solution. This is done to enhance the exploration of the search space by individual chromosomes and to avoid getting stuck in local optima. The two operators considered for this study are the bit flip and inversion mutation. Figure 8 illustrates the working of these operators.

Bit Flip Mutation

The bit flip mutation operator is used for binary represented chromosomes. The operator creates a copy of the parent chromosome and iterates over the array of objects within the chromosome. As the name suggests, the operator flips the binary value stored at each position with mutation probability p_{mf}.



(a) (b)

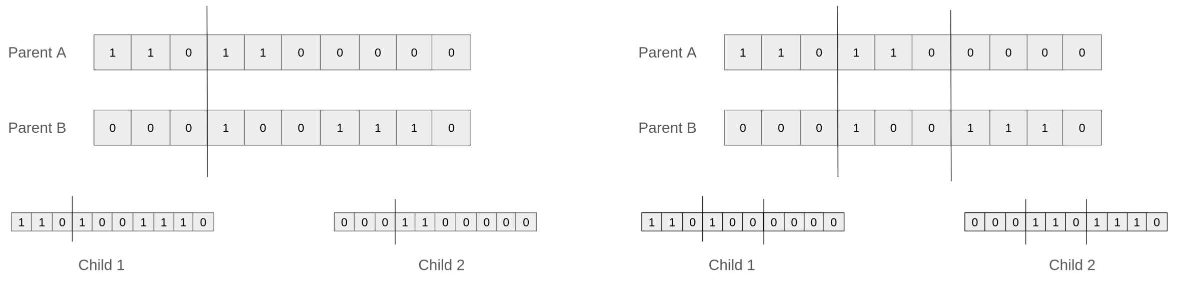
Figure 8: Examples of bit flip(a) and inversion (b) mutation operators.

Inversion Mutation

In inversion mutation, two points in the parent solution are selected randomly. The values of the solution that lie between these two points are then inverted to generate a child solution.

4.4 Crossover

The crossover operator is a probabilistic operator that combines parts of two chromosomes into an offspring. Parts of the parent chromosomes with high fitness are combined to direct the search towards (local) optima. The two crossover operators considered for this study are the one and two-point crossover. Figure 9 illustrates the working of these operators.



(a) (b)

Figure 9: Examples of one-point(a) and two-point (b) crossover operators.

one-point crossover

For one-point crossover in an one-dimensional representation of a chromosome, one location on the array is selected randomly. For the generation of a child chromosome, the values from the start up to the selected location are taken from the first parent and the values from the chosen location up to the end are selected from the second parent.

two-point crossover

For two-point crossover in an one-dimensional representation of a chromosome, two locations on the array are selected randomly. For the generation of a child chromosome, the values from the start up to the first selected location are taken from the first parent, the values from the first location to the second are taken from the second parent, and the values from the second location up to the end are again selected from the first parent.

4.5 Selection and Fitness

4.5.1 Non-Dominated Sorting

The non-dominated sorting is an operator that introduces Pareto optimality into the selection process of NSGA based algorithms. All GAs variants studied in this research use the same non-dominated sorting algorithm. This algorithm iterates through each solution within the population and assigns its rank based on the number of solutions that dominate it. Algorithm 1 shows the pseudocode of the implemented sorting procedure.

Algorithm 1 Non-Dominated Sorting Algorithm

```
1:  **Input:** $L_p$ (Population list)
2:  **Output:** $D_P$ (Dictionary of Pareto fronts with ranks as keys)
3:  $D_P \leftarrow \emptyset$
4:  for all $S \in L_p$ do
5:      $dominated \leftarrow 0$
6:      for all $S' \in L_p$ do
7:          if $S = S'$ then
8:              continue
9:          if $U(S') \leq U(S)$ and $R(S') \leq R(S)$ and $(U(S') < U(S)$ or $R(S') < R(S))$ then
10:             $dominated \leftarrow dominated + 1$
11:     if $dominated$ is a key in $D_P$ then
12:         add $S$ to the set in $D_P$ at entry $dominated$
13:     else
14:         add $dominated$ as key in $D_P$ with entry $\{S\}$
15: Return $D_P$
```

4.6 Crowding Distance

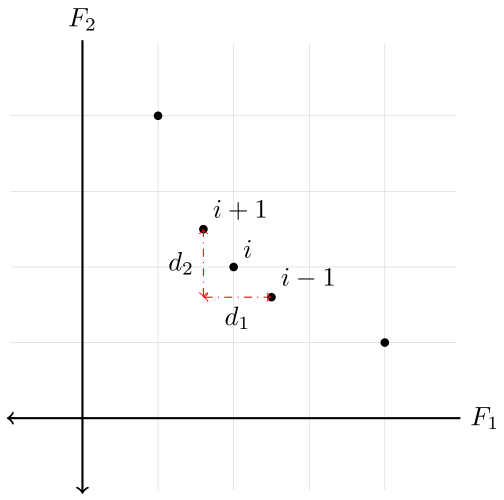


Figure 10: The calculation for the crowding distance of the point i, where d_1 gives the Euclidean distance between the two nearest points of i for minimisation objective F_1 and d_2 for the objective F_2.

A key feature of the NSGA-ii is the utilisation of a crowding distance to order solutions within the same rank. The crowding distance is a metric which gives an indication of how close a solution lies to the other solutions within the same rank. When choosing between solutions with the same rank, the solution with the higher crowding distance is selected to keep the population pool genetically diverse. Algorithm 2 shows the pseudocode for the crowding distance and Figure 10 illustrates the calculation of the crowding distance for a minimisation problem with two objectives.

Algorithm 2 Crowding Distance Assignment (NSGA-II)

```
 1: Input: $front$ (List of solutions on the same Pareto front)
 2: Output: $distance$ (List of crowding distances)
 3: $N \leftarrow$ number of solutions in $front$
 4: $M \leftarrow$ number of objectives
 5: $distance[i] \leftarrow 0$ for all $i = 1$ to $N$
 6: for $m = 1$ to $M$ do                                  ▷ Loop over each objective
 7:     Sort $front$ in ascending order based on objective $m$
 8:     $f_{\min} \leftarrow$ minimum value of objective $m$
 9:     $f_{\max} \leftarrow$ maximum value of objective $m$
10:     $distance[0] \leftarrow \infty$
11:     $distance[N-1] \leftarrow \infty$                   ▷ Boundary solutions get infinite distance
12:     for $i = 1$ to $N-2$ do
13:         if $f_{\max} - f_{\min} \neq 0$ then
14:             $prev \leftarrow$ value of objective $m$ for solution $i-1$
15:             $next \leftarrow$ value of objective $m$ for solution $i+1$
16:             $distance[i] \leftarrow distance[i] + \frac{next - prev}{f_{\max} - f_{\min}}$
17: return $distance$
```

4.6.1 Penalty Functions

The process of generating new solutions is a stochastic process, meaning new solutions can be generated that do not satisfy the constraints of the optimisation problem. There are multiple methods to handle invalid solutions in the search phase of the GA. Here, three of the possible methods will be discussed. The first method to handle invalid solutions is solution reparation, where infeasible solutions are modified such that they become feasible. The solution trimming algorithm applied in the initialisation is an example of solution reparation. This method is acceptable during the initialisation stage, as the solutions are randomly generated at the start. However, a drawback for this method is the loss of genetic information, making it less preferable for the search phase.

The second method is the death penalty function, which disregards infeasible solutions and continues to generate new solutions until enough valid offsprings have been created. A major drawback of this function is that the search process can stagnate if the feasibility region is small [7], as is the case with the minimisation problem defined in section 3.2. Due to the stochastic nature of the offspring creation, it is possible that the process of generating valid offspring from solutions within a population becomes highly improbable as solutions grow towards the limits of the optimisation constraints. To illustrate: solutions with high workloads are (after a few iterations) likely to also be high in urgency. If a solution borders on the maximal capacity, a single addition of an object after the mutation operator can render the solution invalid since each object has a workload of one or more. Analogously, solutions that have a lower workload are more likely to have lower risks values associated, and the deletion of objects will cause the solution to be disregarded, even if the solution only violates the constraint by one or two workload points. And as shown in section 3.3, the optimal solution is likely to lie near or on the boundaries of the given constraints. So the choice to keep on generating new feasible solutions with high probability of failure will lead to high computational costs.

With the drawbacks of previous methods in mind, a static penalty function is a promising operator that retains genetic information whilst applying increasing selective pressure on infeasible solutions by adjusting the fitness of a solution based on its compliance to the constraints set by the optimisation problem [17]. This method is also lower in expected complexity in comparison to the death penalty method for the given problem, as the offspring generation is sure to conclude in reasonable time if the penalty function is applied to each generated offspring regardless of its feasibility. In addition, research has shown that it is preferable to keep infeasible solutions within the population pool for the overall performance of a GA [14, 21]. So for the minimisation problem defined in 3.2.1, a static penalty function $F_p(S)$ can be constructed. Let $\delta(S)$ be defined as the indicator function

$$\delta(S) = \begin{cases} 1 & \text{if } C \geq W(S) \geq C_{min} \\ 0 & \text{otherwise} \end{cases}$$

and

$$d(S) = |W(S) - C + C_{min}|$$

as the difference between the estimated workload of a solution S and the constraints of the minimisation problem. Then, the static penalty function F_p for all objectives is given by

$$F_p(S) = \begin{pmatrix} R(S) \\ -U(S) \end{pmatrix} + \delta(S) \begin{pmatrix} d(S) \\ d(S)^2 \end{pmatrix},$$

where the order of the penalty value $d(S)$ is adjusted to fit the order of magnitudes of the function values. If $d(S)$ is small relative to the value of the objective, infeasible solutions may be included in the final result. Conversely, if $d(S)$ is too large relative to the objective, the algorithm might converge to sub-optimal solutions [17].

4.7 Performance Metric

To evaluate and compare the performance of each GA, a performance metric is needed. For multi-objective evolutionary algorithms, the existing performance metrics mainly measure the algorithms against the convergence to the theoretical optimal Pareto front, the spread of the solution across the search space, or the number of solutions found [13].

For test graph defined in Section 3.3, the exact Pareto front is known, so it is possible to measure the performances of the GAs by their convergence to the exact Pareto front. The General Distance (GD) metric is an existing performance metric that evaluates GAs using the Euclidean distances between the found solutions and the nearest points on the Pareto front [11]. The formula of the GD given by

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

where n is the number of solutions found and d_i is the Euclidean distance between the i-th solution and the nearest point on the exact Pareto front. This metric is chosen due to its simple implementation and relatively low computational cost.

For general graphs to which the algorithm can be applied, the exact Pareto front is unknown. So for the general case, the focus of the performance metric should lie on either the diversity or number of solutions. Here, the choice is made to prioritise the spread of the solutions over the number of solutions, as it is preferable for solution making to have a few diverse options as opposed to having many similar solutions to choose from. The chosen heuristic for the performance metric is therefore constructed as follows: Let $F = \{S_1, \ldots S_n\}$ be the non-dominated solutions found by a GA, and define $F_{\min} = \min\{(U(S_i), R(S_i))|S_i \in F\}$, $F_{\max} = \max\{(U(S_i), R(S_i))|S_i \in F\}$. The performance is calculated as

$$-|F_{\max} - F_{\min}|.$$

This formulation maximises the distance between the outermost solutions on the found Pareto fronts but also allows the hyperparameter optimisation framework SMAC3 (see section 4.8) to define the hyperparameter tuning as a minimisation problem.

4.8 Parameter Fine-tuning

The GAs contain hyperparameters. These are parameters hidden within an algorithm such as the probabilities for the mutation and crossover operators. Hyperparameter optimisation is an essential but complex part of algorithm design, since the chosen values for these parameters can significantly effect the performance of an algorithm. Here, the SMAC3

Table 6: Optimal hyperparameter values found for the test graph, using the non-deterministic SMAC3 optimisation with random seed 0.

	Crossover	Mutation	Bit Flip Rate
One-point & Bit Flip	0.580168	0.062490	0.717704
One-point & Inversion	0.861773	0.341534	–
Two-point & Bit Flip	0.797828	0.6423228	0.132549
Two-point & Inversion	0.921692	0.643638	–

framework is used for the hyperparameter tuning. SMAC3 is a python package that optimises hyperparameters of an algorithm by defining the outputs as results in an minimisation problem and it uses Baysian optimisation techniques to find the optimal set of parameter values [8]. Since GAs are stochastic, the optimisation is performed using the non-deterministic optimisation offered by the framework, with 100 trials per optimisation process. The population sizes and the number of iterations per run are not included in the hyperparameters to be optimised, as those values can have significant impact on the runtime. The choice is therefore made to evaluate those values manually. The optimised hyperparameter values for each GA variant executed on the test graph are shown in Table 6. For the experiments on the GAs performances for larger graphs, a graph is generated randomly following the properties described in Section 3.1. The graph has a total of 1080 nodes, and to simplify the analysis process, all objects within this graph have been given the same workload value and urgency values 1 to 5. The parametric values for a larger general graph with 1080 nodes are shown in Table 7.

Table 7: Optimal hyperparameter values found for a general graph with 1080 nodes, using the non-deterministic SMAC3 optimisation with random seed 0.

	Crossover	Mutation	Bit Flip Rate
One-point & Bit Flip	0.553710	0.804866	0.029642
One-point & Inversion	0.260865	0.111057	–
Two-point & Bit Flip	0.409249	0.887455	0.750594
Two-point & Inversion	0.260865	0.111057	–

5 Results

To evaluate the performance of different GAs, a series of experiments are conducted on both the test graph with 100 nodes as described in section 3.3 as well as on a larger generated graph with 1080 nodes. The general distance metric is used to evaluate the performance of each GA. Experiments on the test graph are performed to test the effects on the performance of GAs due to variations in population sizes and the number of iterations per GA. Experiments on the performance of various GAs are also carried out on a larger bipartite graph with 1080 nodes of general form along with experiments on the effect of the initialisation and trimming function for larger graphs. Unless specified otherwise, all test experimental values are collected from 100 runs per GA and all statistical analyses mentioned are performed using the two-sided Wilcoxen rank-sum test with a confidence level of 95%. The experiments are performed on a workstation with a 13th Gen Intel Core i7-1370P CPU and 32 GB RAM, running on Linux Mint 6.8.0-51-generic.

5.1 Performance Comparison on the Test Graph

The first experiment tests the effect of difference in population sizes on the performance. From the results shown in Table 8, the GA with one-point crossover and inversion mutation at a population size of 200 has the lowest mean GD value. This is supported by statistical analysis on both the comparison between the first and second best performing GA at a population size of 150, as well as a two-sided test for the results of the one-point and inversion variant with a population size of 150 and 200, the second best overall performer. The tests result in p-values of 1.240e-4 and 7.590e-13 respectively. Overall, GAs utilising inversion mutation have lower GD values than the ones using bit flip mutation.

Table 8: Mean GD values of found solutions to the exact Pareto front for each GA and population size. The mean is taken over 100 runs per GA and 250 iterations each run. The overall best score is highlighted.

	50	100	150	200
One-Point, Bit Flip	9.90e-2	5.50e-2	6.40e-2	2.00e-2
One-Point, Inversion	9.35e-3	1.41e-3	3.70e-4	7.54e-4
Two-Point, Bit Flip	8.54e-1	7.69e-3	6.03e-3	1.52e-1
Two-point, Inversion	1.80e-2	4.72e-3	1.48e-3	3.96e-3

As the values for both the mean and the standard deviation for the GD are both small, the choice is made to study the coefficient of variation (CV) as a measure of the relation between the standard deviation and the mean rather than the standard deviation itself. As shown in Table 9 ,all values for the CVs are strictly less than 1 and the standard deviation is relatively small compared to the mean for each performed test. The GAs with the inversion mutation operator has consistently lower valued coefficients than those with bit flip mutation. The GA with the overall best mean GD value has a relatively small CV value, but the second overall best has a lower CV value. All GAs have the lowest CV value at a population size of 100, except for the variant with one-point crossover and bit flip mutation. This GA has its lowest CV value at a population size of 200. Figure 11 shows the performance of the GAs with various population sizes after one run.

Table 9: Coefficients of variation values of the general distance of found solutions to the exact Pareto front for each GA and population size. The CV values are taken over 100 runs per GA and 250 iterations each run. The lowest CV values per GA type are highlighted

	50	100	150	200
One-Point, Bit Flip	2.48e-1	2.21e-1	1.92e-1	6.05e-2
One-Point, Inversion	4.36e-2	6.86e-3	1.62e-2	9.00e-3
Two-Point, Bit Flip	1.92e-1	7.03e-3	1.96e-1	7.69e-2
Two-point, Inversion	1.11e-2	3.48e-3	5.10e-3	1.63e-2

For the mean runtime of the GAs, the values increase strictly along with the population size. This is shown in Table 10. Overall, the variants with one-point crossover perform consistently better than those with two-point

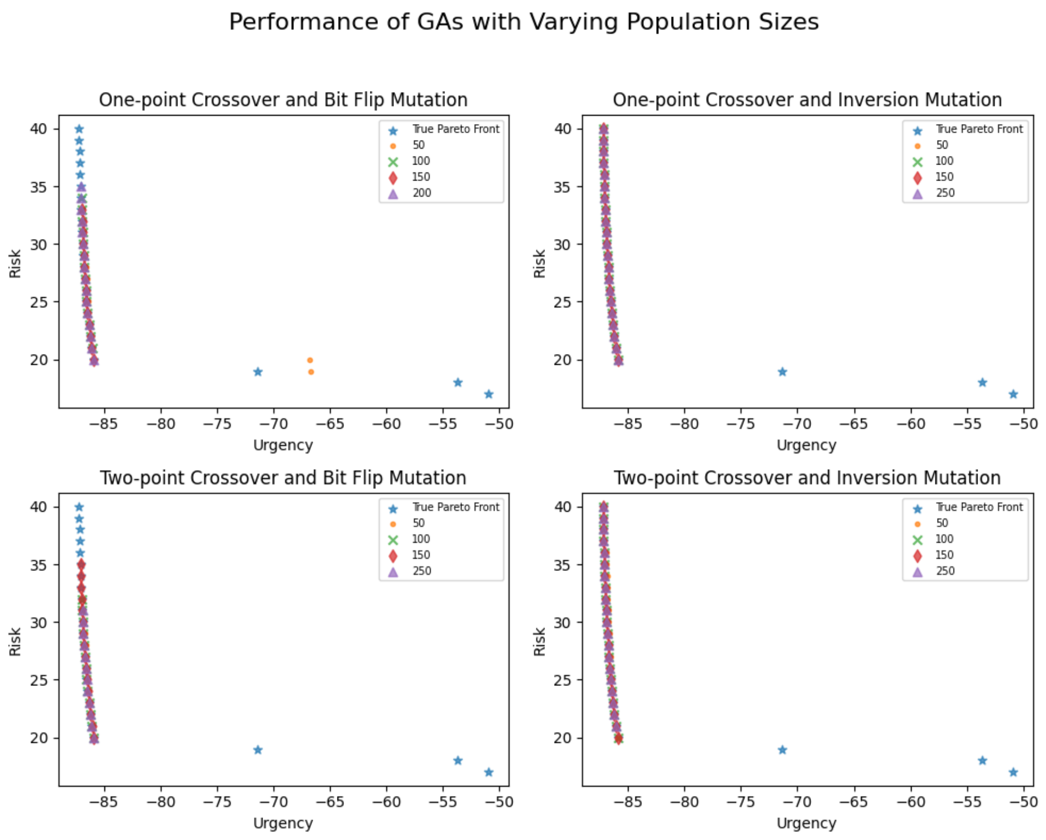


Figure 11: Approximation of the exact Pareto front after one run of each GA for for varying population sizes with 250 iterations per run.

crossover. The average runtimes for the one-point crossover variants are the same for population sizes $\geq$ 150. Statistical analysis on the runtimes result in p-values 0.518 and 0.735 respectively for the difference between the runtimes at population size 150 and 200. So the statistical analysis supports the observation.

Table 10: Mean runtime in seconds for each GA and population size. The mean is taken over 100 runs per GA and 250 iterations each run. The best runtime per population size is highlighted.

	50	100	150	200
One-Point, Bit Flip	9.24e-1	1.24	1.61	2.01
One-Point, Inversion	9.88e-1	1.27	1.61	2.01
Two-Point, Bit Flip	1.28	1.58	1.95	2.33
Two-point, Inversion	1.01	1.29	1.63	2.03

The second experiment is performed by varying the number of iterations per run. The resulting GD values are shown in Table 11. Again, the GAs with one-point crossover and inversion mutation perform the best, with the closest approximation at 300 iterations. Statistical analysis is performed on the comparison between the first and second best performing GA at 300 iterations, as well as a two-sided test for the results of the one-point and inversion variant with 150 and 300 iterations. The tests result in p-values of 4.31e-2 and 9.80e-16 respectively for a confidence level of 95%. So the statistical analysis supports the observation that the GA with one-point crossover, inversion mutation at 300 iterations performs better than the other variants. One run of all GAs for each number of iterations is shown in Figure 12.

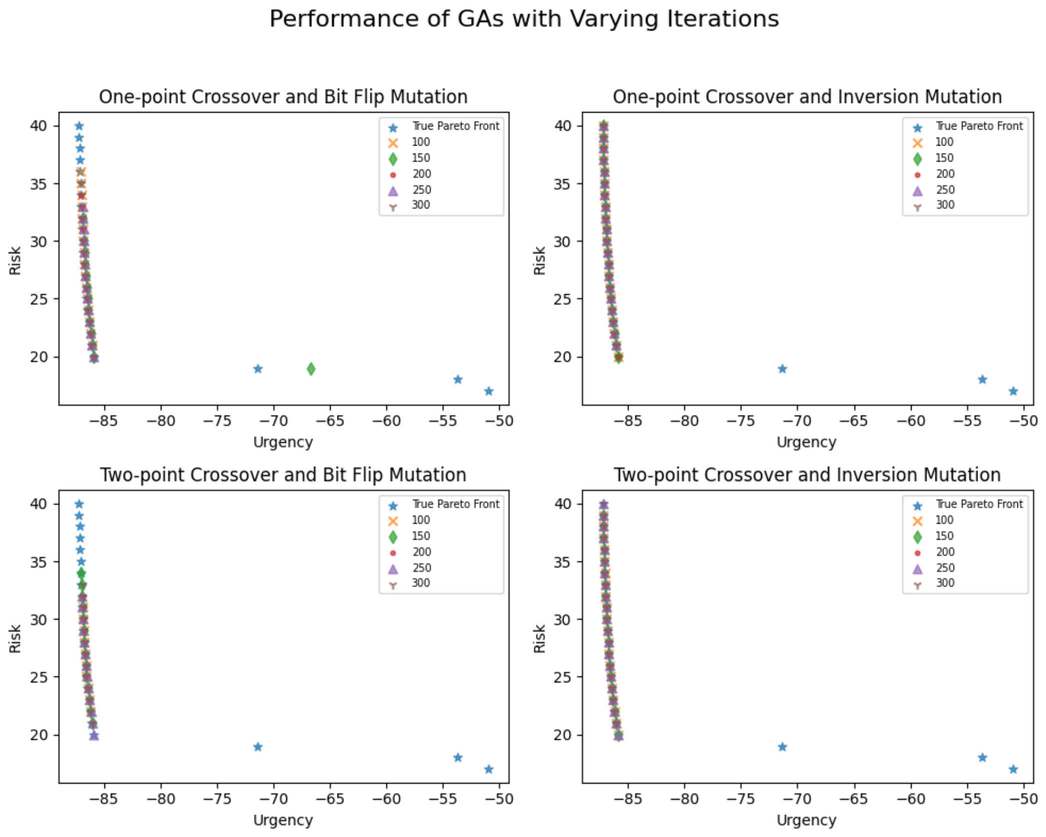


Figure 12: Approximation of the exact Pareto front after one run of each GA for varying numbers of algorithm iterations at a population size of 100.

Analysis on the coefficient of variation of the found GD values for varying number of iterations are shown in Table 12. The CVs are the highest for the variant with two-point crossover and bit flip mutation and the lowest for the variant with one-point crossover and inversion mutation. The coefficient of variation is significantly higher for GAs which employ bit-flip mutation than inversion mutation. The coefficient values are also higher for GAs that use two-point crossovers rather than one-point crossover. The lowest CV values for all GA variants are found at 300 iterations, except for the variation with one-point crossover and bit flip mutation, whose lowest value lies at 150 iterations. However, the CV value for this GA variant at 300 iterations is found to be very close to the lowest CV value ($< 5.0\text{e-}5$ difference).

Table 11: Mean general distance to the exact Pareto front for each GA with varying numbers of iterations per algorithm. The mean is taken over 100 runs per GA, each with a population size of 100. The overall lowest mean GD value is highlighted.

	100	150	200	250	300
One-Point, Bit Flip	9.15e-1	6.57e-1	7.18e-1	5.86e-1	7.37e-1
One-Point, Inversion	4.55e-3	1.75e-3	3.28e-3	2.76e-3	1.39e-3
Two-Point, Bit Flip	2.61	1.99	1.46	1.74	1.31
Two-point, Inversion	3.30e-2	1.50e-2	9.83e-3	4.19e-3	2.96e-3

The average runtime for the GAs increases roughly proportionately with the number of iterations, as shown in Table 13. The GA with the one-point crossover and bit flip mutation performs the best across all number of iterations, with the one-point and inversion mutation variant as a close second. Statistical analysis determines that the variants with one-point crossover are not statistically equal in runtime. With a p-value of 0.0253 on a confidence level of

Table 12: Coefficient of variation for each GA and varying numbers of iterations per algorithm. The coefficients are calculated from 100 runs per GA, each with a population size of 100. The lowest CV value for each GA variant is highlighted.

	100	150	200	250	300
One-Point, Bit Flip	1.08	7.42e-1	8.80e-1	6.63e-1	6.47e-1
One-Point, Inversion	1.11e-2	6.82e-3	1.00e-2	7.63e-3	6.86e-3
Two-Point, Bit Flip	2.92	3.37	2.55	2.33	1.30
Two-Point, Inversion	6.59e-2	4.35e-2	4.78e-2	1.44e-2	1.16e-2

95%, the null-hypothesis is rejected. Statistical analysis is also performed to compare the runtimes of the GAs with inversion mutation, as their runtime averages are similar. From the analysis, it follows that only the results of the the runs with 100 and 200 iterations are statistically similar, with p-values of 0.060 and 0.051 respectively. The null-hypothesis is rejected for the tests with other number of iterations. The GA with two-point crossover and inversion mutation scored the worst across all iterations.

Table 13: Mean runtime in seconds for each GA and number of iterations. The mean is taken over 100 runs per GA, each with a population size of 100.

	100	150	200	250	300
One-Point, Bit Flip	5.20e-1	7.70e-1	1.02	1.28	1.50
One-Point, Inversion	5.22e-1	8.05e-1	1.03	1.28	1.53
Two-Point, Bit Flip	6.65e-1	9.98e-1	1.30	1.60	1.90
Two-point, Inversion	5.23e-1	7.78e-1	1.04	1.30	1.56

5.2 General Graph

Figure 13 shows the fronts found by the GAs after one run. Each GA has a population size of 100 and 250 iterations per run. The graph has 324 cross environment connections in total, this is indicated by a horizontal line in Figure 13.

The mean performance values of these GAs over 100 runs are shown in Table 14. The GA variant with two-point crossover and bit flip mutation operator has the highest mean runtime of all, followed by the other variant with bit flip mutation. The variants with inversion mutation operators have the lowest mean runtime. These variants also find more than double the number of solutions compared to those with bit flip inversion operators. Table 15 show that the the standard deviation of the number of solutions found by the GA with one-point crossover and bit flip mutation is larger than its corresponding mean value, indicating a high level of variance. The standard deviation of the GA with two-point crossover and bit flip mutation also has a relatively high standard deviation compared to the mean value, but it remains less than the mean value. The GAs with inversion mutation have much smaller values for the standard deviation, but only the variant with two-point crossover has a standard deviation values that is less than 1.

Table 14: Mean values of the performance values found per GA on a graph with 1080 nodes. The mean is taken over 100 runs per GA, each with a population size of 100 and 250 iterations.

GA	Runtime	Solutions	New Risks	Urgency
One-Point, Bit Flip	16.173	14.840	-20.660	121.850
One-Point, Inversion	7.732	36.570	176.774	13.899
Two-Point, Bit Flip	28.748	4.140	25.210	52.469
Two-Point, Inversion	7.724	36.840	177.180	13.924

When studying the mean values found for the number of introduced risks and total urgency, Table 14 shows that only the GA variant with one-point crossover and bit flip mutation decreases the total risk level of the graph, but it comes with a high standard deviation value, as seen in Table 15. In addition, this is also the variant that finds solutions with the highest mean level of urgency. In contrast, the variants with inversion mutation find solutions that

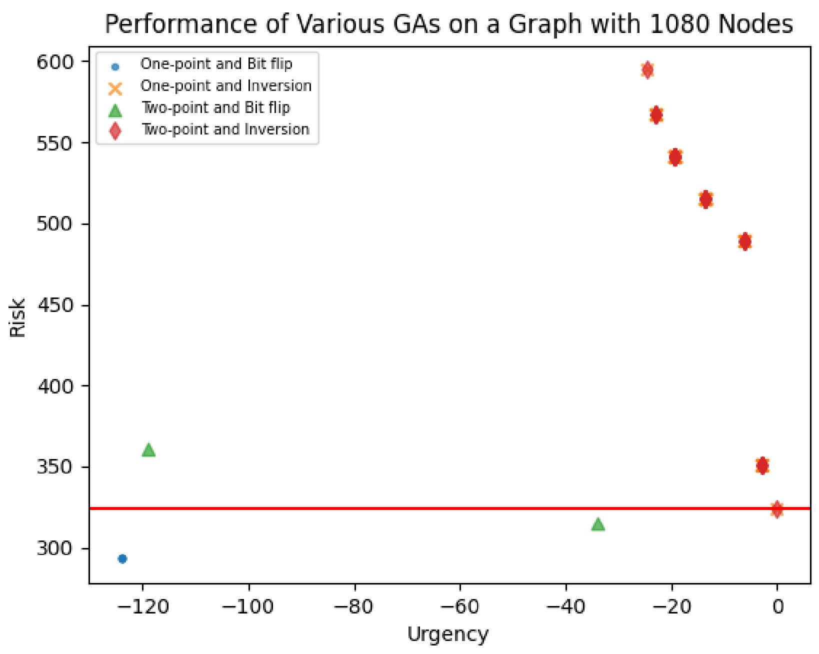


Figure 13: The Pareto fronts found by each GA after one run. All GAs have a population size of 100 and perform 250 iterations per run.

introduce the highest mean number of risk while also having the lowest level of urgency. The performance of the GA with two-point crossover and bit-flip mutation falls somewhere between the others. The values found for the GAs with inversion mutation operators are very similar across the tests. Statistical analysis on the comparison between the runtime, number of solutions, and average distance of the front to the starting risk line showed that the runtimes and level of urgency of the two are statistically equal (p-values of 0.7985 and 0.03768 respectively), but that the number of solutions, average number of new risks were not; with p-values of 1.377e-3 and 1.965e-4 respectively on a 95% confidence level. Further statistical analysis reveals that both the values for the number of risks and solutions are lower for the GA with two-point crossover and inversion mutation than its one-point counterpart. Overall, the values found in Table 14 reflect the run plotted in Figure 13, where it is clear that the GA variant with the one-point crossover and bit flip mutation dominates all fronts found by the other GAs.

Table 15: Standard deviation of the performance values found per GA on a graph with 1080 nodes. The values are calculated from 100 runs per GA, each with a population size of 100 and 250 iterations.

GA	Runtime	Solutions	New Risks	Urgency
One-Point, Bit Flip	0.591	17.348	30.279	9.589
One-Point, Inversion	0.110	1.565	2.721	0.216
Two-Point, Bit Flip	0.151	2.353	24.048	33.103
Two-Point, Inversion	0.0565	0.368	1.460	0.062

The results of further analysis on the composition of found solutions are shown in Tables 16 and 17. In these Tables, it is shown that the GA with one-point and bit flip mutation has a strong bias towards solutions containing almost exclusively application objects. Closer inspection reveals that many of the found solutions are similar, often with just one or two applications that are different. In contrast, the GAs with inversion mutation exclusively find solutions with management objects. Only the variant with two-point crossover and inversion mutation finds solutions that generally include both types of objects, but the bias towards application object is still evident. Furthermore, the GAs with bit flip mutation operators have relatively high standard deviation values for the number of management objects in their solutions, especially compared the the standard deviation value of the number of application objects in the solutions of their counterparts with inversion mutation operators.

Table 16: Mean values of the solution compositions found for each GA on a graph with 1080 nodes. The mean is taken over 100 runs per GA, each with a population size of 100 and 250 iterations.

GA	AO	MO	Workload
One-Point, Bit Flip	30.169	0.313	30.481
One-Point, Inversion	0.0	6.497	6.497
Two-Point, Bit Flip	12.753	1.177	13.930
Two-Point, Inversion	0.0	6.499	6.499

Table 17: Standard deviation of the solution compositions found per GA on a graph with 1080 nodes. The values are calculated from 100 runs per GA, each with a population size of 100 and 250 iterations.

GA	AOs	MOs	Workload
One-Point, Bit Flip	3.500	1.099	2.575
One-Point, Inversion	0.0	0.113	0.113
Two-Point, Bit Flip	6.919	0.738	6.871
Two-Point, Inversion	0.0	0.049	0.049

5.3 Trimming function

To test the effect of the trimming function on the overall performance of the GAs, the GAs are run both with and without the trimming function during initialisation. Table 18 shows the comparison in runtime for both variants. As seen in the table, the runtimes are very similar. Statistical analysis on the differences in runtimes all result in p-values less than 0.05 for runtimes except for the GA variant with one-point crossover and bit flip mutation, which has a p-value of 0.0656 on a confidence level of 95%.

Table 18: Mean values and standard deviation of the runtime per GA on a graph with 1080 nodes with trimming (left) and without (right). The values are calculated from 100 runs per GA, each with a population size of 100 and 250 iterations.

GA	Runtime		Standard Deviation	
One-Point, Bit Flip	16.138	16.259	0.566	0.593
One-Point, Inversion	7.752	7.695	0.0715	0.0518
Two-Point, Bit Flip	28.780	28.829	0.169	0.129
Two-Point, Inversion	7.739	7.665	0.053	0.0593

Tables 19 shows the comparison between the performance of the two variant in terms of the number of new risks introduced and the number of solutions. Table 20 shows the corresponding standard deviation values. The found values for the mean number of introduced risks show that all GAs perform better with the trimming function, except for the variant with two-point crossover and bit flip mutation operator. The latter however, is combined with a high level of variance, as the corresponding standard deviations for this variant are both larger than the mean values. For the mean urgency of the solutions found, the value lessens for the variants with bit flip mutation, but increases for those with inversion mutation. As for the mean urgency, this value increases for the GAs with bit flip mutation when the trimming function is applied but decreases for those with inversion mutation. Lastly, for the effect of the trimming function on the number of solutions, Table 19 shows that the number of solutions decreases for the bit flip variants but increases for those with inversion mutation when the trimming function is applied.

Table 19: Comparison of mean values of performance measures found for GAs with trimming (left) and without (right) per GA on a graph with 1080 nodes. The mean is taken over 100 runs per GA, each with a population size of 100 and 250 iterations.

GA	New Risks		Urgency		Solutions	
One-Point, Bit Flip	-25.043	-22.095	121.850	118.943	14.470	15.790
One-Point, Inversion	176.740	182.998	13.899	14.677	36.720	34.510
Two-Point, Bit Flip	22.588	13.070	52.469	50.479	4.000	4.750
Two-Point, Inversion	176.843	183.117	13.924	14.740	36.910	34.950

Some of the standard deviation values shown in Table 20 are relatively large compared to their corresponding mean value. A Statistical analysis on the differences in the number of introduced risks result in p-values of 0.535 for the variant with one-point crossover and bit flip mutation and 2.20e-16, 0.00312 and 2.20e-16 respectively for the other 3 variants in order of the table. Tests on the differences in urgency reveal that the null-hypothesis cannot be rejected for the GAs with bit flip mutation, as they have p-values of 0.130 (one-point) and 0.503 (two-point) in comparison with the p-values and 2.20e-16 for both variants of the GAs with inversion mutation. Lastly, tests on the mean number of solutions found result in p-value 0.433 for the GA with one-point mutation, 0.0290 for its two-point crossover counterpart and again 2.20e-16 for both variants of the GAs with inversion mutation.

Table 20: Comparison of standard deviation values of performance measures found for GAs with trimming (left) and without (right) per GA on a graph with 1080 nodes. The mean is taken over 100 runs per GA, each with a population size of 100 and 250 iterations.

GA	New Risks		Urgency		Solutions	
One-Point, Bit Flip	20.008	28.194	9.589	16.851	19.047	24.071
One-Point, Inversion	2.955	3.689	0.216	0.155	1.471	2.0225
Two-Point, Bit Flip	22.585	21.067	33.103	35.508	2.601	2.393
Two-Point, Inversion	1.005	1.178	0.0618	0.0449	0.287	0.219

6 Discussion

The experiments on the test graph show that the GAs perform well for the given task, as they all have at least one configuration of population size and number of iteration which manages to approximate the analytical Pareto front. The coefficient of variation presented in Table 9 for the tests on population size show that the variations in standard deviation values are small in comparison with the mean GD values, indicating that GAs are able to consistently approximate the exact front when the population sizes are varied. The values shown in Table 12 indicate that the same holds for the tests on the varying number of iterations, with the exception of the GA with two-point crossover and inversion mutation, whose large CV value infer varying performance for the Pareto front approximation.

Moreover, Table 8 shows that for varying population sizes, the GA with one-point crossover and inversion mutation approximate the exact Pareto front the best overall at a population size of 150. The GAs using inversion mutation yield closer approximations to the exact Pareto front described in Section 3.3 for all population sizes and number of iterations per run compared to those with bit flip mutation. The GAs with the bit flip mutation operators yield solutions that approximate the lower ends of the optimal front more often than their inversion mutation counterparts, as illustrated in Figure 11 and 12.

In terms of runtime, the GA variant with the one-point crossover and bit flip mutation operator performed the best on the test graph regardless of the population size and number of iterations, as shown in Tables 10 and 13. This is followed by the runtimes of GAs with inversion mutation, whose runtime results lie closely together across all tests. The GA variant with two-point crossover and bit flip mutation has the highest overall runtime. This is a trend that is continued when the GAs are applied to a much larger test graph with 1080 nodes. As shown in Table 14, the mean runtime for the GA with two-point crossover and bit flip mutation is almost twice the mean runtime of its one-point counterpart and nearing four times the mean runtime of the GAs with inversion mutation. Furthermore, the increase in time complexity is much larger in the variants with the bit flip operator than the ones with inversion mutation. This could be explained by the structure of the operators themselves, as the bit flip operator needs to iterate over each element in the chromosomes, meaning that the complexity grows proportionately to the product of the number of nodes, the population size, and the number of iterations. In comparison, as the region of inversion is determined stochastically, the number of operations that the inversion performs on average does not grow as fast. This could make the GAs bit flip mutation a bad choice terms of time complexity if the number of nodes grows to be larger. However, in real life scenarios where the algorithm would be applied, the number of nodes is limited by the physical storage capacity and will not increase significantly more than the size of the graph with 1080 upon which the tests are conducted. Thus the runtime of all tested GAs will still remain within reasonable boundaries for the designed application.

Another observation about the differences in runtime between the GAs, is that the complexity of the bit flip operator does not seem to explain the large difference in runtime between the one-point and two-point crossover variants with the bit-flip operator, as the difference in runtime between the GAs with inversion mutation are almost equal. However, it is also important to note that the hyperparamter values found by SMAC3 for the two GAs with inversion mutation operators are identical, while the values for all hyperparamters for the GAs with bit flip mutation operators are different, as shown in Table 7. This could also have impacted the overall test results.

Furthermore, no exact reason can be derived from the test results in Section 5.2 as to why the GA variant with inversion mutation perform worse on larger graphs compared to those with bit flip operator. A potential explanation could be that the inversion mutation is too volatile at the start, as larger inversions can drop or include large parts of objects with closely related ID numbers, which would coincide with application objects connected to the same management object due to the method of the landscape generation. This mutation operator is therefore able to drastically change the fitness of a solution within one generation if the inversion region is large. Thus solution with relatively less application elements are more robust against these drastic changes. This idea is supported by the composition of the solutions found by the GAs with inversion mutation shown in Table 16, as those contain no application objects and only have few management objects. The robustness is reflected in the low standard deviation values shown in Table 17. Statistical analysis shows that the variant show that the crossover also have an effect on the performance of the GA, as the version with two-point crossover performed slightly better than the one with one-point crossover. Further research on the performance of these GAs on graphs with varying structures such as low and fixed number of application objects connected to management objects or having few management objects and larger number of application objects is needed to validate these interpretations.

In contrast to the GAs with inversion mutation, the GA with one-point crossover and bit flip mutation has a strong bias towards solutions that contain solely application objects, as shown in 16. The GA generally finds many solutions with the same number of risks and urgency. These solutions are often copies of each other with the exception of one or two different application elements. This could be explained by the properties of the graph itself, as all objects within the graph have been given the same workload but varying urgencies (1 to 5). If solutions do not contain management objects, the difference in risks and urgency is lower for solutions. For example, both the risk and the urgency remains the same after switching one application object in the lower security level with another that has the same urgency level. The creation of these similar solutions with small variation are more likely to be created by this GA, especially since the actual probability of a bit flip is low ($\sim$0.03), as seen in Table 7. And when the found solutions have (mostly) converged to similar solutions, the effect of the crossover operator lessens. While this could explain the reason the algorithm creates many similar solutions on the graph used for the experiments, the found solution composition cannot be established as a definitive pattern or behaviour characteristic of the GA in general. Further experiments on the performance of this GA on graphs with varying properties is needed to research the types of solution composition found by this GA.

The GA with two-point crossover and bit flip mutation is the only GA that contains both types of objects in its solutions, but it also has a bias towards solutions with more application objects. The performance of this variant falls between the three other GAs, meaning that the fronts it finds dominate those of the GAs with inversion mutation, but is dominated by the variant with one-point crossover and bit flip mutation. This GA is also the variant with the lowest mean number of optimal solutions found, as shown in 14. This could be explained by its much higher bit flip rate ($\sim$0.75) compared to the one-point crossover variant, making this particular bit flip operator very volatile.

From the experiment on the effect of the trimming function on the overall performance of the GAs, it seems that the effect of the trimming function is minimal. Table 18 shows that the additional time complexity of the function is negligible for the graph with 1080 nodes. Statistical analysis performed on the results in Table 19, show that no clear effects of the trimming function on the GA with one-point crossover and bit flip mutation can be discerned. However, the GAs with inversion mutation find more solutions with lower risks levels when the trimming function is applied. This does slightly decrease the urgencies of the solutions found. For the GA with two-point crossover and bit flip mutation, the trimming function has an overall negative impact. No effects on the mean urgency of solution can be observed for when the trimming function is applied, but it leads to more risks and a lower number of solutions.

Finally, the experiments performed in this paper show that it is possible to construct suitable GAs for the creation of automated and optimised LCM backlogs. In addition, as the selection algorithm of the GAs is based on Pareto optimality, the GAs can easily be adapted to include more properties and expand the dimensionality of the search space if the need arises. On a more abstract level, IT landscapes with communicating entities such as the one studied here can generally be well represented through graphical representations. This study shows that genetic algorithms can be applied to strategic (hardware) migration in these environments. Moreover, the GAs can be used within a larger computational framework to automatically and iteratively generate and select optimal solutions show possible future configurations of the application landscape to offer strategic business insights on the state of affairs. This gives rise to possibilities of future studies on the applicability and effectiveness of algorithms as part of decision making and strategic planning in growingly larger and more complex environments.

7 Conclusion

In this paper, the problem of constructing an automated strategic LCM backlog is addressed for the complex application landscape of SSC-ICT. Four NSGA based genetic algorithms have been proposed to solve the two-objective optimisation problem. A mathematical framework is constructed to translate the application landscape to a graphical representation. To measure the performances of the GAs, a theoretical landscape with 150 objects and an exact Pareto front is constructed. To simulate the behaviour of the GAs for a realistic environment, a larger landscape with 1080 objects is generated.

The results from experiments on the theoretical landscape show that the proposed GAs are all able to approximate the analytically determined Pareto front. However, the performance starts to vary greatly when the GAs are applied to the larger, more complex graph with 1080 nodes. The experiments show that the GA variants using inversion mutation perform worse than other variant. They return similar results across all tests and find solutions with a strong bias for management objects. Statistical analysis shows that the GA with two-point crossover performs slightly better, making the GA with one-point crossover and inversion mutation the worst overall performer. The second best performing GA variant is the one with two-point crossover and bit flip mutation. It finds solutions that generally contain both application and management objects, but has the highest runtime of all GAs. Finally, the experiments show that the GA variant with one-point crossover and bit flip mutation performs the best and consistently finds Pareto fronts that dominate those found by the other GAs. Even though it has a bias for solutions that contain only application objects, this GA is the only variant that finds solutions that lower the overall risk level in the environment.

References

[1] Carlos A. Coello. “An updated survey of Ga-based multiobjective optimization techniques”. In: *ACM Computing Surveys* 32.2 (June 2000), pp. 109–143. DOI: 10.1145/358923.358929.

[2] K. Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[3] Yujia Ge and Guiyi Wei. “Ga-Based Task Scheduler for the Cloud Computing Systems”. In: *2010 International Conference on Web Information Systems and Mining* (Oct. 2010), pp. 181–186. DOI: 10.1109/wism.2010.87.

[4] Nyoman Gunantara. “A review of multi-objective optimization: Methods and its applications”. In: *Cogent Engineering* 5.1 (2018). Ed. by Qingsong Ai. DOI: 10.1080/23311916.2018.1502242. eprint: https://doi.org/10.1080/23311916.2018.1502242. URL: https://doi.org/10.1080/23311916.2018.1502242.

[5] J.H. Holland. “Genetic algorithms”. In: *Scientific American 267(1), 66–73* (1992).

[6] Arvind Kumar, Reetika Nagar, and Anurag Singh Baghel. “A genetic algorithm approach to release planning in Agile Environment”. In: *2014 International Conference on Information Systems and Computer Networks (ISCON)* (Mar. 2014), pp. 118–122. DOI: 10.1109/iciscon.2014.6965230.

[7] Konstantinos Liagkouras and Konstantinos Metaxiotis. “Examining the effect of different configuration issues of the multiobjective evolutionary algorithms on the efficient frontier formulation for the constrained portfolio optimization problem”. In: *Journal of the Operational Research Society* 69.3 (Dec. 2017), pp. 416–438. DOI: 10.1057/jors.2016.38.

[8] Marius Lindauer et al. “SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization”. In: *Journal of Machine Learning Research* 23.54 (2022), pp. 1–9. URL: http://jmlr.org/papers/v23/21-0888.html.

[9] H. Ma, Y. Zhang, S. Sun, et al. “A comprehensive survey on NSGA-II for multi-objective optimization and applications”. In: *Artificial Intelligence Review* 56 (2023), pp. 15217–15270. DOI: 10.1007/s10462-023-10526-z.

[10] mcdowella. *Complexity theory - polynomial time algorithm for set partition on powers of 2? - stack overflow.* Dec. 2014. URL: https://stackoverflow.com/questions/27222058/polynomial-time-algorithm-for-set-partition-on-powers-of-2.

[11] Seyedali Mirjalili and Andrew Lewis. “Novel performance metrics for robust multi-objective optimization algorithms”. In: *Swarm and Evolutionary Computation* 21 (2015), pp. 1–23. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2014.10.005. URL: https://www.sciencedirect.com/science/article/pii/S2210650214000777.

[12] João Luiz Pereira et al. “A review of multi-objective optimization: Methods and algorithms in mechanical engineering problems”. In: *Archives of Computational Methods in Engineering* 29.4 (Oct. 2021), pp. 2285–2308. DOI: 10.1007/s11831-021-09663-x.

[13] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. “Performance metrics in multi-objective optimization”. In: *2015 Latin American Computing Conference (CLEI)*. 2015, pp. 1–11. DOI: 10.1109/CLEI.2015.7360024.

[14] D. Ansa Sekharan and Roger L. Wainwright. “Manipulating subpopulations of feasible and infeasible solutions in genetic algorithms”. In: *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing states of the art and practice - SAC ’93* (1993), pp. 118–125. DOI: 10.1145/162754.162831.

[15] Saltuk Buğra Selçuklu et al. “Multi-objective Genetic Algorithms”. eng. In: *Handbook of Formal Optimization.* Singapore: Springer Nature Singapore, 2024, pp. 1007–1044. ISBN: 9789819738199.

[16] Shubhkirti Sharma and Vijay Kumar. “A comprehensive review on multi-objective optimization techniques: Past, present and future”. In: *Archives of Computational Methods in Engineering* 29.7 (July 2022), pp. 5605–5633. DOI: 10.1007/s11831-022-09778-9.

[17] Alice E. Smith and David W. Coit. *Section C 5.2 of Handbook of Evolutionary Computation.* Oxford University Press and Institute of Physics Publishing, 1995.

[18] N. Srinivas and Kalyanmoy Deb. “Muiltiobjective optimization using nondominated sorting in genetic algorithms”. In: *Evolutionary Computation* 2.3 (Sept. 1994), pp. 221–248. DOI: 10.1162/evco.1994.2.3.221.

[19] T. Tan K.and Lee and Khor E. “Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons.” In: *Artificial Intelligence Review 17, 251–290* (2002). URL: https://doi.org/10.1023/A:1015516501242.

[20] Zitong Wang, Yan Pei, and Jianqiang Li. “A survey on search strategy of evolutionary multi-objective optimization algorithms”. In: *Applied Sciences* 13.7 (Apr. 2023), p. 4643. DOI: 10.3390/app13074643.

[21] Jiawei Yuan, Hai-Lin Liu, and Zhaoshui He. “A constrained multi-objective evolutionary algorithm using valuable infeasible solutions”. In: *Swarm and Evolutionary Computation* 68 (2022), p. 101020. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2021.101020. URL: https://www.sciencedirect.com/science/article/pii/S2210650221001826.