electronics

MDPI

Article

A Framework for Component Selection Considering Dark Sides of Artificial Intelligence: A Case Study on Autonomous Vehicle

Mohammad Reza Jabbarpour [1,*], Ali Mohammad Saghiri [1] and Mehdi Sookhak [2]

[1] Information and Communications Technology Research Group, Niroo Research Institute, Tehran 1468613113, Iran; amsaghiri@nri.ac.ir
[2] School of Information Technology, Illinois State University, Normal, IL 61761, USA; msookha@ilstu.edu
* Correspondence: mrjabbarpour@nri.ac.ir

Abstract: Nowadays, intelligent systems play an important role in a wide range of applications, including financial ones, smart cities, healthcare, and transportation. Most of the intelligent systems are composed of prefabricated components. Inappropriate composition of components may lead to unsafe, power-consuming, and vulnerable intelligent systems. Although artificial intelligence-based systems can provide various advantages for humanity, they have several dark sides that can affect our lives. Some terms, such as security, trust, privacy, safety, and fairness, relate to the dark sides of artificial intelligence, which may be inherent to the intelligent systems. Existing solutions either focus on solving a specific problem or consider the some other challenge without addressing the fundamental issues of artificial intelligence. In other words, there is no general framework to conduct a component selection process while considering the dark sides in the literature. Hence, in this paper, we proposed a new framework for the component selection of intelligent systems while considering the dark sides of artificial intelligence. This framework consists of four phases, namely, component analyzing, extracting criteria and weighting, formulating the problem as multiple knapsacks, and finding components. To the best of our knowledge, this is the first component selection framework to deal with the dark sides of artificial intelligence. We also developed a case study for the component selection issue in autonomous vehicles to demonstrate the application of the proposed framework. Six components along with four criteria (i.e., energy consumption, security, privacy, and complexity) were analyzed and weighted by experts via analytic hierarchy process (AHP) method. The results clearly show that the appropriate composition of components was selected through the proposed framework for the desired functions.

Keywords: intelligent systems; component selection; learning automata; autonomous vehicle

Citation: Jabbarpour, M.R.; Saghiri, A.M.; Sookhak, M. A Framework for Component Selection Considering Dark Sides of Artificial Intelligence: A Case Study on Autonomous Vehicle. *Electronics* **2021**, *10*, 384. https://doi.org/10.3390/electronics10040384

Communicated by: Michele Segata
Received: 14 December 2020
Accepted: 29 January 2021
Published: 4 February 2021

1. Introduction

At present, there are many studies on component-based software architecture, integration and selection; architecture mismatch analysis; and off-the-shell (OTS) based development in the literature. Initially, software component composition is introduced in [1] as "an assembly of parts (components) into a whole (a composite) without modifying the parts." It means the components should be composed in a way to satisfy functional requirements, where each component contains a clearly defined interface and functional description. The architecture-centric manner is utilized in traditional component selection (CS) approaches in which the most suitable available alternative component in the market is selected by considering the description of the desired component.

Various commercial OTS (COTS) approaches have been proposed in the literature, including [2–5]. In [2,3], the authors focused on COST selection and implementation by considering the business and functional criteria. However, architectural constraints without considering interoperability issue proposed in [4] as COST selection. A time-consuming and manual approach for COST component interoperability evaluation was proposed in [6], which is not suitable for evaluating a large number of COST combinations.

The second type of CS methods are designed by considering the relationship between requirements and available components for use. The main goal of these methods is to identify the mutual affection between requirements and available components to acquire a set of requirements, which is consistent with the desired market goal. The requirements for engineering and CS methods are incorporated to achieve this goal. PORE and CRE are the two most notable examples of the second type of CS methods [5]. It is not realistic to anticipate a completed match between desired components and available components in any CS approach. A set of components that creates a system may overlap and have gaps in required functionality. A gap represents a lack of functionality, while an overlap can lead to responsibility confusion and declines in non-functional attributes, including performance and size.

A report given in [7] by Gartner shows that 59% of organizations are going to build their own intelligent systems (ISs). This report also reveals that leading organizations are going to increase their numbers of projects, which involve IS features and components. ISs have been widely used in different domains, such as healthcare [8], financial management [9], tourism [10], information systems [11], transportation [12], autonomous vehicles [13], and marketing [14]. Although artificial intelligence (AI) methods have various dark sides, including high energy consumption and a high pollution rate, many studies have focused on their utilization to improve the resource management, efficiency, and accuracy of decision making processes [15] in different contexts. Despite the existence of a few studies, such as [16], on the dark sides of AI in designing ISs, this problem has not been considered in designing component-based software. It is obvious that inappropriate composition of components may lead to developing an unsafe, power-consuming, and vulnerable IS. As a result, it is essential to use CS to engineer the dark sides of AI in ISs.

CS methods have evolved considering the substantial challenges associated with intelligent systems, including dark sides of artificial intelligence. A list of potential AI dark sides is as follows:

- Energy consumption [17].
- Data issues [18,19].
- Security and trust [20].
- Privacy [21].
- Fairness [22,23].
- Safety [24,25].
- Beneficial [26].
- Predictability [27,28].
- Explainable AI [29].
- The complexity issue [30].
- Monopoly.
- Responsibility challenges [31].

Section 4.2.1 explains such issues in more detail. Some of the above challenges were used in the literature to define new versions of CS in different contexts, such as electric vehicles (EVs) [32], smart buildings [33], and renewable energy systems [34]. These studies only focused on challenges without considering information about AI-based components. Therefore, information about AI-based components does not affect the component selection process. On the other hand, some studies, such as [35–37], present mechanisms that consider some dark sides of AI such as privacy and data corruption, but they propose ad hoc solutions. In other words, they do not develop a general solution for developing software considering the dark sides of AI. Therefore, the existing solutions are not general enough to be used in different domains. In the next paragraph, the context of EVs is studied in more detail.

CS has also been considered in EVs, wherein there are different components, such as electric motors, power converters, and energy storage systems, which play a critical role in EVs' architectures. A comparative study on various components of EVs is presented in [32]. Finding the best coordination among the existing components can lead to the incrementing

of traveled distance and fuel consumption reduction by EVs. Moreover, the use of dynamic programming in the CS process was proposed in [38] to reduce EVs' fuel consumption. On the other hand, control theory and learning algorithms are widely used in designing EVs. As some recent works, we can refer to [39–41]. However, all of the available algorithms suffer from a critical problem, which is the lack of a general framework to develop AI-based systems considering dark sides of AI.

In this paper, we developed a new framework to overcome the component selection problem of ISs considering the dark sides of AI. The proposed framework consists of four phases, namely, component analyzing, extracting criteria and weighting, formulating the problem as multiple knapsacks, and finding components. After describing and analyzing components' attributes via experts in the first phase, dark sides of AI techniques are extracted in second phase through a comprehensive study. Moreover, the AHP method is utilized to compute appropriate weights for components in this phase. In the third phase, the CS problem by considering obtained weights is formulating as a multiple knapsack problem, which is solved by using the learning automata algorithm in the last phase. The main contributions of this paper are as follows:

- Discussion of the general concepts of CS problem and its variations;
- An extensive and comprehensive study on the dark sides of AI techniques to extract the main technical dark sides;
- Proposing a novel framework for the CS problem of ISs that considers the dark sides of AI.

The organization of the paper is as follows: Related works are discussed in Section 2. Section 3 discusses the learning automata theory as a solution for the knapsack problem. Section 4 is dedicated to the proposed framework's explanation, including its phases. The applicability of the proposed framework was investigated through a case study, i.e., an autonomous vehicle, in Section 5. Section 6 is dedicated to managerial implications of the proposed framework. Finally, Section 7 concludes the paper and suggests the direction for future research.

2. Related Works

Several researchers have been focused on the complexity of CS approaches. For example, CS is defined as the problem of selecting the minimum number of components from a group of components so that their combination meets a set of goals [42]. It is an NP-complete optimization problem formally showed to be embedded within compensability [43]. This problem is similar to the cardinality set cover problem where all components have equal costs. Various genetic and heuristic algorithms have been used to solve this problem. However, a different definition is presented in [44] for the CS problem as the process of modeling an engineering system from OTS components by merging them to form a functional system. For this purpose, there are some generic components and each component can be implemented via a set of components. This problem is defined as the process of choosing particular components from producer's catalog by considering components' mutual effects on each other. The genetic algorithm is utilized to solve this problem in [44]. Two algorithms are proposed in [45] for the component section problem and the next release problem. The next release problem deals with the selection of a subset of requirements based on their desirability. In this problem, each component is determined via a set of values including anticipated development duration, revenue, and cost.

Although one of the well-known NP-hard problems is the knapsack problem, it can be solved by a pseudo-polynomial algorithm using dynamic programming. Another definition of CS is presented in [46], where they proved it as an NP-complete problem.

Recently, a fuzzy-based approach is proposed in [47] for CS by considering both functional and non-functional requirements. The fuzzy clustering groups similar components at each selection step based on the desired requirements. Finally, the authors developed a reservation system on the basis of different requirements to effectively evaluate the proposed model. In [48] a scorecard-based CS method is proposed based on high-level quality

attribute indicators, project health measures, and a context-specific aggregation function to provide an explicit decision (yes or no) for integrators. Kaur et al. [49] developed a software CS architecture on the basis of the clustering concept. The proposed architecture includes four tiers: component requirements and selection, query and decision, application with clustering, and component clustering. In this architecture, components are clustered based on user defined requirements and their similarities. As a result, the search space, time, and cost of the CS procedure are reduced. According to the exploratory analysis, cost, support for the component, longevity prediction, and level of off-the-shelf fit to a product are the four main attributes of components for industry practitioners in CS [50].

In addition to software engineering, CS has been used in different domains. For example, in [33] smart buildings that are equipped with sensors, devices, and automation systems of the smart building have been considered as a very complex component selection task due to the variety of available vendors and technologies. Hence, to decrease the exponential complexity of the CS problem and minimize the search space, the authors developed a multi-step method for analyzing the prioritized criteria.

In the power industry, the CS plays an important role in the performance of the off-grid system. In [34], the CS for renewable energy systems has been developed based on the total cost and power reliability of the system. The authors leveraged the selection of renewable energy system components to electrify the rural area used for evaluation purposes. Moreover, a multi-variable linear regression in a company with the gradient descent algorithm was used for impact assessment. This approach outperforms the existing methods in terms of optimum selection of components for an off-grid systems.

3. Learning Automata Theory

A field of AI called reinforcement learning focuses on designing ISs that are able to learn optimal policy for decision making in an unknown environments. Among reinforcement learning algorithms, Q-learning and learning automata are well-known. The main difference between these models is that Q-learning invests in the changes of the states in the environment of the learning system, but learning automata theory does not invest in the information of changes related to the environment. Therefore, in some problems reported in engineering and mathematics claiming that formalizing information about the environment is either expensive or impossible, the theory of learning automata can be suitable. In the rest of this section, the theory of learning automata and its applications are explained. In the theory of learning automaton (LA), an intelligent agent is able to make a decision with a self-adaptive manner decision making model [51]. The learning process of this model is made through repeated interactions with a random environment (Figure 1). In the theory of learning automata, the learning automata can be classified into two classes: fixed and variable structure [52]. A variable structure LA can be represented by a triple $< \beta, \alpha, L >$ where β is the set of feedbacks of the environment, α is the set of actions, and L is the learning algorithm. Let $\alpha_i(k) \in \alpha$ and $P(k)$ denote the action selected by LA and the probability vector defined over the action set at instant k, respectively.

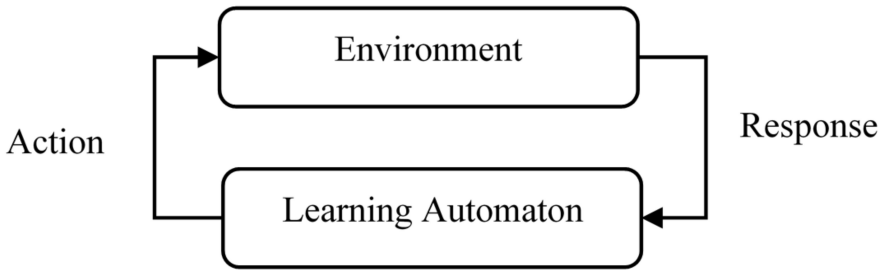

Figure 1. Learning automaton and the environment.

The a and b indicate the reward and the penalty parameters, and r indicates the number of actions that can be selected by LA. At step n, the action is selected based on the action probability vector. Then, the action probability vector $P(n)$ is updated by the linear learning algorithm given in Equation (1), if the selected action $\alpha_i(n)$ is rewarded by the environment, and it is updated as given in Equation (2) if the taken action is penalized. If $a = b$, the recurrence Equations (1) and (2) are called the linear reward penalty (LRP) algorithm. More information can be found in [52]:

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \quad \forall j, j \neq i \end{aligned} \tag{1}$$

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \tfrac{b}{r-1} + (1-b)p_j(n) \forall j, j \neq i \end{aligned} \tag{2}$$

In recent years, LA has been used in different applications, such as cognitive networks [53], computer networks [54], and the shortest-path problem [55] to mention a few.

4. Proposed Framework

We designed a new framework in which a process to select components for ISs considering the dark sides of AI is given. ISs that are considered in this section can be self-driving cars, intelligent drones, and industrial robots, but they are not limited to a specific domain. In this framework, a set of components are provided to design a software package based on the component-oriented design. Table 1 introduces the symbols that are used to formally explain the phases of the proposed framework.

This framework consists of four phases as described below and illustrated in Figure 2.

- **Component analyzing:** according to the information about the component set (CO) and attributes of the components, F set and appropriate components for each function are determined by the expert.
- **Extracting criteria and Weighting:** in this phase, we extract the required criteria on the basis of the dark sides of AI to organize the AHP procedure and compute the weight for each component.
- **Formulating the problem as multiple knapsacks:** in this phase, the problem of CS is mapped to multiple knapsack problems based on the generated weights in the previous phase.
- **Finding components:** In this phase, learning automata theory is used to solve the knapsack problem, and the results are utilized to determine the S set. It is worth noting that there is a possibility to apply different solver algorithms in this phase of the proposed framework.

Figure 3 shows an example of the essential components in autonomous vehicles [12]. This example illustrates how component-based software is able to manage different parts of autonomous vehicles. In this example, finding appropriate components considering safety, privacy, and energy consumption that are relevant criteria to the dark sides of AI is a challenging problem. This example will be simplified and used as a case study to show the applicability of the proposed framework in Section 4.

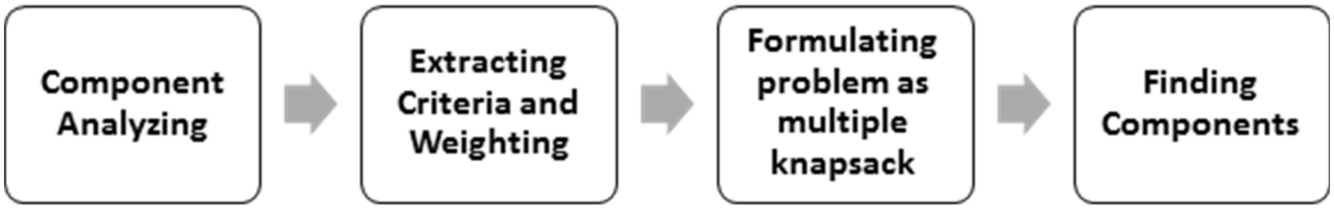

Figure 2. Phases of the proposed framework.

Table 1. Symbols of the framework.

Symbol	Definition
$CO = \{CO_1, CO_2, \cdots, CO_n\}$	Refers to a set of all components that can be used in the final package of the software.
$F = \{f_1, f_2, \cdots, f_q\}$	Refers to a set of functions that should be handled by components. For example F = {Object Detection, Function Approximation}
z_{f_i}	Refers to a set of components that are appropriate for function f_i
K_i	Refers to knapsack i
S	Refers to a set of 3-tuple that are selected for constructing the final package of software. e.g., (K_a, f_b, CO_c) can be a member of this set
$CR = \{cr_1, cr_2, \cdots, cr_r\}$	Refers to a set of all criteria
w_i^{cr}	Represents weight of component i considering criterion cr
a_{ij}	Determines the comparison result between $criterion_i$ and $criterion_j$
m	Refers to the number of knapsacks which is equal to the number of components that should be selected through the component selection procedure
c_i	Refers to capacity of knapsack i
x_{ij}	Refer to select of i^{th} component for j^{th} function
n	Refers to the number of components
p_i	Refers to the profit of component i
v_i	Refers to the total weight of component i considering all criteria

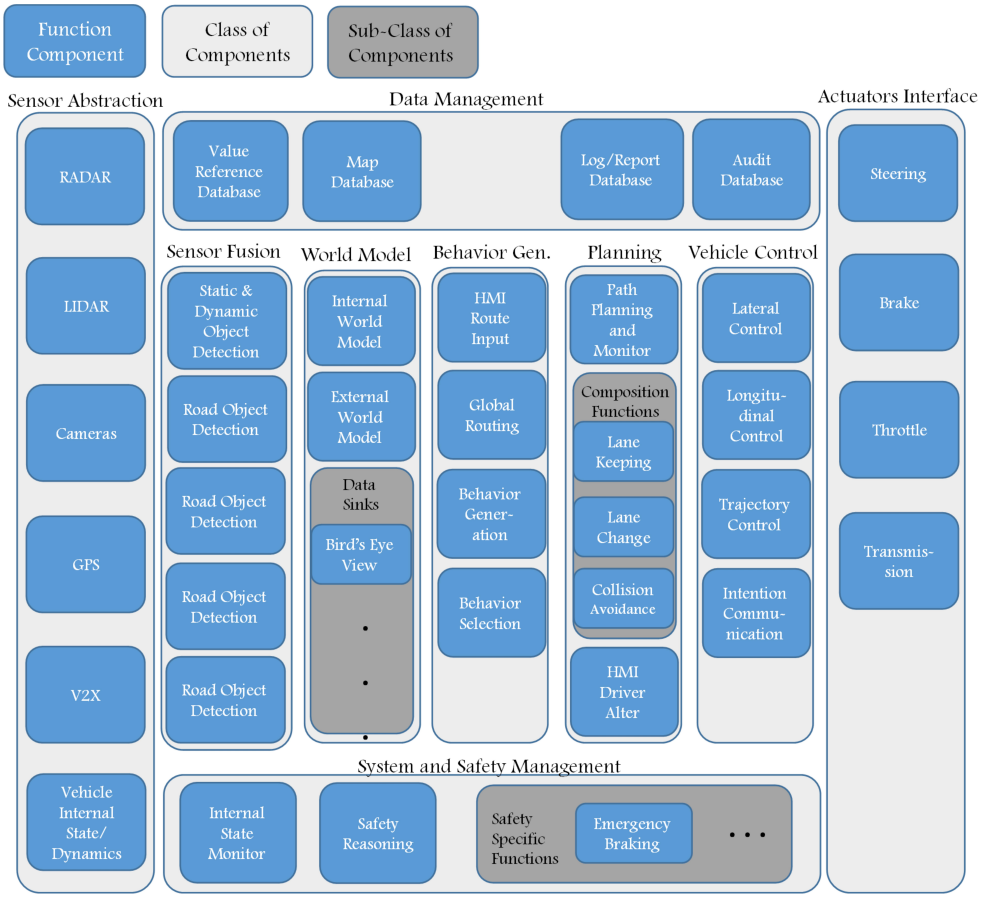

Figure 3. Components for autonomous vehicle [12].

Detailed explanations about phases are given in the rest of this section.

4.1. Component Analyzing

In the component analyzing phase, we generate the F set, which contains appropriate components for each function along with the corresponding attributes for each component.

4.2. Extracting Criteria and Weighting

In this phase, some criteria considering the dark sides of AI are extracted and then the AHP mechanism is used to calculate the weights of components. This phase is organized into two sub-phases as follows.

- Extracting criteria: wherein about 20 papers focusing on the dark sides of AI are reviewed and then 12 criteria are extracted (refer to Section 4.2.1 for more details).
- Weighting: during this sub-phase, the AHP mechanism is organized to find the weights of components considering criteria extracted in the previous sub-phase (refer to Section 4.2.2 for more details).

In the next two sub-sections, the above sub-phases are explained in more detail.

4.2.1. Extracting Criteria

Although AI provides significant changes and improvements for networked digital business and facilitates smart services and digital transformation, there are plentiful dark sides of AI that present tremendous risks for individuals, organizations, and society. To address these dark sides, the first and most important step is to identify and classify such criteria. Therefore, a list of potential AI dark sides is as follows:

- Energy consumption.
- Data issues.

- Security and trust.
- Privacy.
- Fairness.
- Safety.
- Beneficial.
- Predictability.
- Explainable AI.
- Complexity issue.
- Monopoly.
- Responsibility challenges.

The above terms are explained in more detail in the rest of this section.

1. **Energy Consumption.** One of the dark sides of most machine learning algorithms is high energy consumption. Nowadays, the majority of machine learning algorithms rely on iterative policies instead of fixed policies, which leads to high energy consumption and energy wastage. Moreover, this problem is accelerated by a growing number of learning models that require more iterations for learning purposes. For example, deep learning methods require the high computational power of GPUs more than other methods. Environmental pollution and global warming are reported as other side effects of high computational power usage [17].
2. **Data Issues.** A category of AI invests in data-driven algorithms to construct machine learning models. It should be noted that in many situations, there are many problems in data that lead to many difficulties in data-driven machine learning. Some of these problems are as follows.
    - Big Data: The size of data sets gathered in a wide range of systems such as IoT and AR is increasing. With a huge amount of data, defining machine learning algorithms that are able to operate in an online fashion leads to a challenging problem [18]. We can use wide range of methods, such as sampling, distributed processing, and parameter estimation to obtain required information from data.
    - Data incompleteness: Incomplete data refers to a challenging problem in the machine learning algorithms. Incomplete data in every data set may mislead the algorithm to learn inappropriate models. This challenge creates uncertainties during data analysis if we do not consider incomplete data during the data analysis step. Many imputation methods exist for this purpose. An initial approach is to fill a training set with the most frequently observed values or to build learning models to predict possible values for each data field, based on the observed values of a given instance [18].

    Other issues in this domain are data heterogeneity, data insufficiency, data uncertainty, data originality, data inaccurateness, imbalanced data, data dynamicity, and high dimensional data [18,19,56,57].
3. **Security and trust.** Security and trust are two important issues that have received much attention in recent years. In ISs, these issues have two dimensions as follows. Most of the papers [58,59] only focused on utilizing ISs to design secure systems. It is worth noting that every piece of software, including ISs, may be hacked or cracked. These issues are not considered by AI experts because the development of ISs in critical systems is in the early stages. For example, in data-driven machine learning, we trust data and a model will be constructed based on it. When data are untrusted, the machine learning model is also untrusted. During the software lifecycle, a malicious person may swap trusted data with untrusted data, and this phenomenon may occur in every data-driven approach. An emerging field called adversarial machine learning was the first attempt to solve some security problems in data-driven machine learning [20]. For other AI-based methods such as genetic algorithms, we may consider attack mechanisms to manipulate the evolutionary processes.

4. **Privacy.** The privacy issue in AI has different dimensions. In recent years, many ISs have been constructed based on big data analyses, data sciences, and data-driven methods. All of these methods are fed by the data of a huge number of users. During the execution of these methods, three different roles are possible, as explained below.
   - A role for data imputation and data owners (or contributors).
   - A role for data analyzers and model manipulators.
   - A role for result visualizers.

   Usually, a programmer has all those roles during designing ISs. However, in industrial projects, the mentioned roles may be played by different entities and these entities may not be trusted considering privacy issues. In order to solve this problem, many efforts have been made by researchers. Federated learning is one of these efforts [21]. Federated learning is a machine learning technique that trains an algorithm across multiple decentralized computers, without exchanging data among them, thereby allowing one to address critical issues such as data privacy, data security, and data access rights.
5. **Fairness.** In AI, a given algorithm is said to be fair if its results are independent of some variables, such as gender, ethnicity, and sexual orientation. The rationale behind this issue is that many people have disabilities and gender must not add rights to users. This issue is recently reported as a hot topic in machine learning, and papers such as [22,23] are reported in the literature depicting modified machine learning algorithms (removing bios considering special fields) while considering this issue. This issue becomes more challenging when some features such as gender and race are sensitive in the culture of humans. Therefore, well-known companies such as IBM, Facebook, and Google are going to introduce a machine learning library considering this issue.
6. **Safety.** AI has shown to be successful algorithms at smart managing systems. In mission-critical, real-world environments, there is little tolerance for failure that can cause damaging effects on humans and devices. In these environments, current approaches are not sufficient to support the safety of humans. Considering this issue, two main approaches are reported in the literature. In the first approach proposed in [24,25], a system is defined to control the output of an IS while considering the safety of humans. In the second approach proposed in [60], computation in the internal parts of an intelligent agent will be manipulated using some weights considering the safety of humans.
7. **Beneficial.** In the near future, ISs will make better decisions than humans in many domains, including computer vision. An IS can diagnose many diseases using image processing techniques that are more efficient than human vision. For more than one decade, humans have trained to extend the works of their ancestors in simple domains such as piping and constructing a city. These jobs can be easily done by ISs and these systems will control many things in the near future. In some situations, an IS may decide to do an action that is harmful for humans. Most of the existing systems with deterministic decision-making mechanisms may easily execute harmful decisions without considering human preferences. In these situations, beneficial AI computation can be applied. With this theory, a system is designated to behave in such a way that humans are satisfied with the results. In these systems, the agent is initially uncertain about what the preferences of humans are, and human behavior will be used to extract information about human preferences [26].
8. **Predictability.** One of the most important issues in designing ISs is predictability. This issue becomes more challenging when many management algorithms in different fields utilize ISs. In designing ISs, many factors exist which destroy the predictability capability. Some of these factors are explained as follows. Paradox and ambiguity are two factors that exist in image, voice, and text, and therefore a system with these types of inputs cannot present a predictable output. Some theorems in computer sciences,

such as Turing undecidability, the Gödel theorem, and the strange loop theorem are used to prove unpredictable behavior in most of the systems [27,28,61].

9. **Explainable AI.** Explainable AI refers to AI methods such that the results of the solution obtained by them can be understood by human experts [29]. Many of the AI-based systems that are known as best problem solvers, such as deep learning, only focus on a mathematically-specified goal system determined by the designers [62]. Therefore, the output of the system may not be understood by a human agent. This problem can be very challenging in military services and healthcare because the rationale behind a decision should be evaluated by a human agent. In the literature, there are some efforts to solve this problem [63].
10. **Complexity.** The complexity of ISs is increasing day by day. The primary versions of ISs invest in a limited set of solutions to do their jobs and therefore their complexity is limited to simple algorithms. Nowadays, the existing ISs utilize numerous learning algorithms. The complexity related to the size and format of data is discussed in Section 4.2. In addition to the mentioned issues, many challenges related to the complexity of input, computation, memory, and output in the ISs are reported in the literature [30]. Novel approaches to solve complex problems in complex systems rely on digital twin technologies [64].
11. **Monopoly.** Many AI-based solutions require huge computational power. There are a few companies (IBM, Amazon, and Microsoft) and countries which invest in AI and high computational power devices. For example, a few countries are pioneers in quantum computation, which is one of the enablers of artificial general intelligence and super-intelligence. This capability may lead to the appearance of a monopoly in the scope of AI. Those companies which can execute many AI-based algorithms are able to do many valuable activities such as developing new drugs and treatments for diseases.
12. **Responsibility Challenges.** AI-based systems, such as self-driving cars, will act autonomously in our world. In many fields, ISs will make better decisions than humans eventually. A challenging question in these systems such as self-driving cars is: who is liable if a self-driving car is involved in an accident? This problem has many dimensions. It seems that many laws must be defined considering those ISs that are involved in decision making processes. From an algorithmic perspective, frameworks will be needed to extract the responsibility of all entities that are involved in decision making processes. In [31], some interesting points related to responsibility issues are covered for a specific case study.

4.2.2. Weighting

In this phase, for each set of components that are suitable for a function, weights are computed. Decision-making in CS can be defined as a problem of multi-criteria decision analysis by considering the dark sides of AI. In this section, the analytic hierarchy process (AHP) is utilized to compute each component's weight for the desired function. Since AHP considers decision maker's subjectivity in determining the preferences for evaluation objectives and also there is a correlation between the criteria, the AHP method is utilized as weighting method for the proposed case study. The AHP can be used when the decision-making process is faced with several options and decision criteria. The desired criteria can be quantitative and qualitative. A similar approach is utilized in [65] for mobile robot path planning. AHP uses a hierarchy structure in which the desired problem is located at the top; the criteria and the solution alternatives are located at the intermediate and bottom levels, respectively. The AHP procedure can be summarized as the following steps:

- **Step 1**— Problem hierarchy: A hierarchy by considering the desired goal (at the top layer), criteria (at the intermediate level), and solution alternatives (at the bottom level) is created. Each criterion can be divided into sub-criteria based on the requirements. The criteria are used by decision makers to set priorities.

- **Step 2**—Set priorities for each criterion: The decision maker assigns a numerical value to each criterion based on preferences. These numerical values can be assigned based on a scale that is presented in Table 2. This scale is proposed in [66] and its effectiveness has been validated by multiple researchers. Paired comparison is performed by the decision maker to set priority by assigning desired weights. For this purpose, a paired comparison matrix should be created in which $a_{ij}(w_i^{cr}/w_j^{cr})$ determines the comparison result between $criterion_i$ and $criterion_j$. It is worth noting that consistency ratio should be computed for each paired comparison matrix in order to prove its consistency and its value should be less than 0.1. The procedure for computing the consistency ratio (CR) is as follows. Equation (3) is used to calculate the consistency index (CI) in which n and λ_{max} denote the number of criteria and eigenvalue of the pairwise comparison matrix, respectively.

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{3}$$

  After calculating the CI value, Equation (4) is utilized to compute the CR in which the random consistency (RC) index is determined by Table 3.

$$CR = \frac{CI}{RC} \tag{4}$$

  Detailed information regarding consistency ratio computation can be found in [67].
- **Step 3**—Define priorities for solution alternatives: this step is the same as step 2, but the paired comparison should be performed between solution alternatives to create preferences based on predefined criteria.
- **Step 4**—Calculate the final priority of solution alternatives: the total weight for the criteria and solution alternatives is calculated from the multiplication of the local weight by the total weight of the immediately superior criterion. The totality of the final weights of the solution alternatives is computed with respect to each criterion.

Table 2. Pairwise comparison scale [66].

Intensity of Importance *	Definition	Explanation
1	Equal importance	Two elements (i, j) contribute equally to the objective
3	Moderate importance	Experience and judgment slightly favor one element over another
5	Strong Importance	Experience and judgment strongly favor one element over another
7	Very strong importance	One element is favored very strongly over another, it dominance is demonstrated in practice
9	Extreme importance	The evidence favoring one element over another is of the highest possible order of affirmation

* 2,4,6,8 can be used to express intermediate values.

Table 3. Random consistency index.

Number of Criteria	RC
3	0.58
4	0.90
5	1.12
6	1.24
7	1.32

Table 3. *Cont.*

Number of Criteria	RC
8	1.41
9	1.45

4.3. Formulating Component Selection as a Multiple Knapsack Problem

In this phase, the CS problem is considered as a $0-1$ multiple knapsack problem (MKP). MKP is a strongly NP-hard combinatorial optimization problem that has many applications, such as resource allocation, financial planning, stock allocation, and shipment loading. MKP includes m knapsacks with capacities of $c_1, c_2, \cdots, c_m$ and n items (components) ($m \leq n$). i^{th} item has an associated profit, p_i and occupies v_j value (total weight) of m_j knapsack. The goal is to fill knapsacks with a subset of items so that the maximum profit is achieved, and the total weight of a backpack′s items does not exceed its capacity. The MKP can be defined as follows:

$$Maximize \ \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} \tag{5}$$

$$Subject\ to \ \sum_{j=1}^{n} v_j x_{ij} \leq \ c_i, \ i \ \in 1 \ldots m \tag{6}$$

$$\sum_{i=1}^{m} x_{ij} \ \leq 1, \ j \ \in 1 \ldots n \tag{7}$$

$$x_{ij} \ \in \ \{0, \ 1\}, i \ \in M, \ \ j \ \in N \tag{8}$$

$$Where \ \ x_{ij} \ = \begin{cases} 1 & if\ item\ j\ is\ selected\ to\ knapsack\ i; \\ 0 & otherwise. \end{cases} \tag{9}$$

It is also assumed that

$$v_j \ \leq \ max_{i \in M}\{c_i\} \quad for \quad j \in N \tag{10}$$

$$c_i \ \geq \ min_{j \in N}\{v_j\} \quad for \quad i \in M \tag{11}$$

$$\sum_{j=1}^{n} v_j \ > \ c_i \qquad for \quad i \in M \tag{12}$$

4.4. Finding Components: A Learning Automata-Based Solution for the Component Selection

In this phase, in order to find the solution to the knapsack problem which formulated in the previous phase, a learning automata-based algorithm is utilized. This algorithm is reported in [68]. This algorithm gets iteration number (Itr), reward (a), and penalty (b) parameters and then searches with a solution space-based probabilistic search mechanism to find an appropriate solution. In this algorithm, the problem is modeled as a complete graph where each node of the graph corresponds to a component in the knapsack problem. Each node of the graph is equipped with a learning automaton with two actions of selecting either item to be placed in a knapsack or not. In this mechanism, agents are defined to activate learning automata. In each iteration of the algorithm, there are a few agents, each of which creates a solution. Initially, an agent is randomly placed on one of the graph nodes and activates the learning automaton of that node. Whenever a learning automaton is activated, it selects one of its two actions according to the probability vector of its actions. Afterward, the solution set is constructed based on the action chosen by the learning automaton. Finally, the solution set is modified considering some criteria. This process is repeated based on the predefined iteration number. The flowchart of this algorithm is represented in Figure 4.

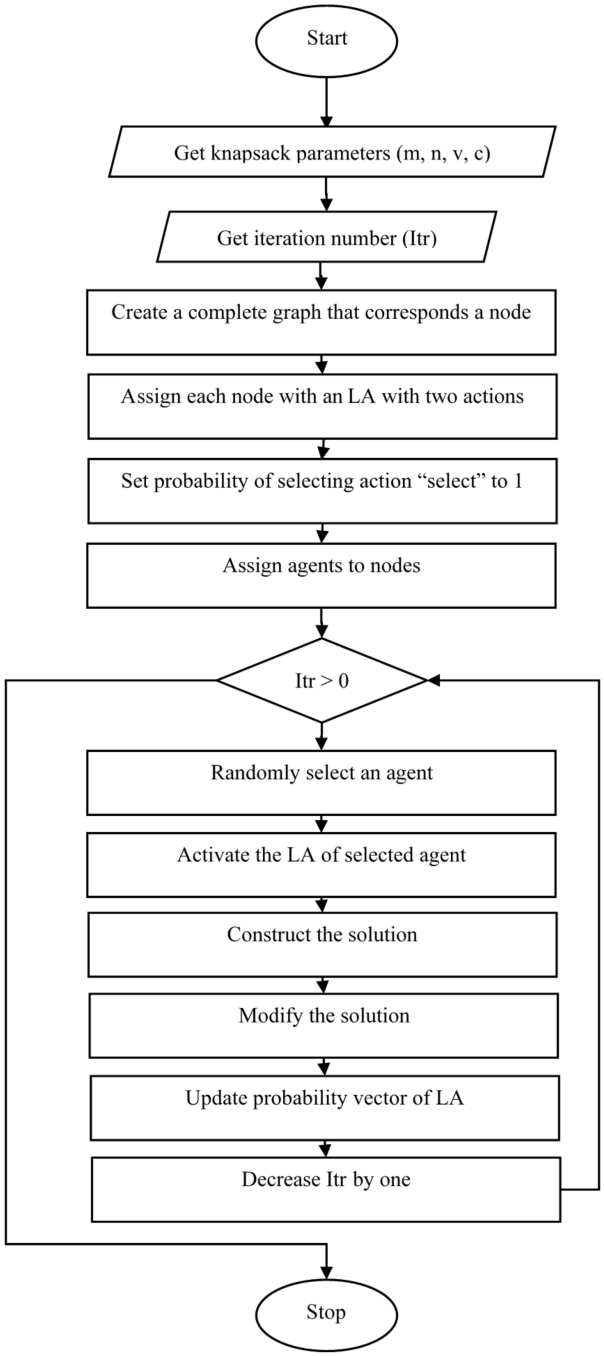

Figure 4. Flowchart of finding a component phase.

Remark 1. *Although in this phase, we have used the learning automata approach to solve the knapsack problem, every other mechanism that is able to solve this problem can be used in this framework as an alternative mechanism.*

Remark 2. *The proposed method has four phases. Phases 1 and 3 will be done by a designer. In these phases, some simple decisions are made by designers, and therefore computational complexity is not a critical matter in them. Computational complexities of other parts are studied as below.*

- *In phase 2, executing AHP may be considered in the computational complexity of the proposed method. Time complexity of AHP is* $O(min\{mn^2, m^2n\})$ *[69], in which m is an alternative and n is a criterion.*

- *In phase 4, an iterative learning method is used to find the solution. This algorithm takes parameter Itr and uses a loop to find the corresponding solution. The computational complexity of this part is* $O(Itr)$.

Therefore, the time complexity of the proposed method is $O(max\{O(Itr), O(min\{mn^2, m^2n\})\})$.

Remark 3. *The proposed method presents a systematic approach for constructing AI-based software considering the dark sides of AI. The proposed method is based on a well-known software engineering perspective that is component based systems. This method is more generalized than ad-hoc solutions such as those reported in [70] that only invest in solving a specific problem.*

5. Case Study

In this case study, autonomous vehicles are considered as well-known ISs. Autonomous vehicles are designated based on a fusion of an IS into a car with sufficient sensors and actuators. As reported in [12], the design of autonomous vehicles can be done using component oriented design, and therefore the problem of CS in this domain is a well-defined problem for the proposed framework in this paper. Figure 5 is an abstraction of Figure 3 that shows the functional components of an autonomous vehicle in which our selected components for this case study are shown in gray. It is desired to select appropriate components for road object detection, dynamic object detection, global routing, and path planning and trajectory control.

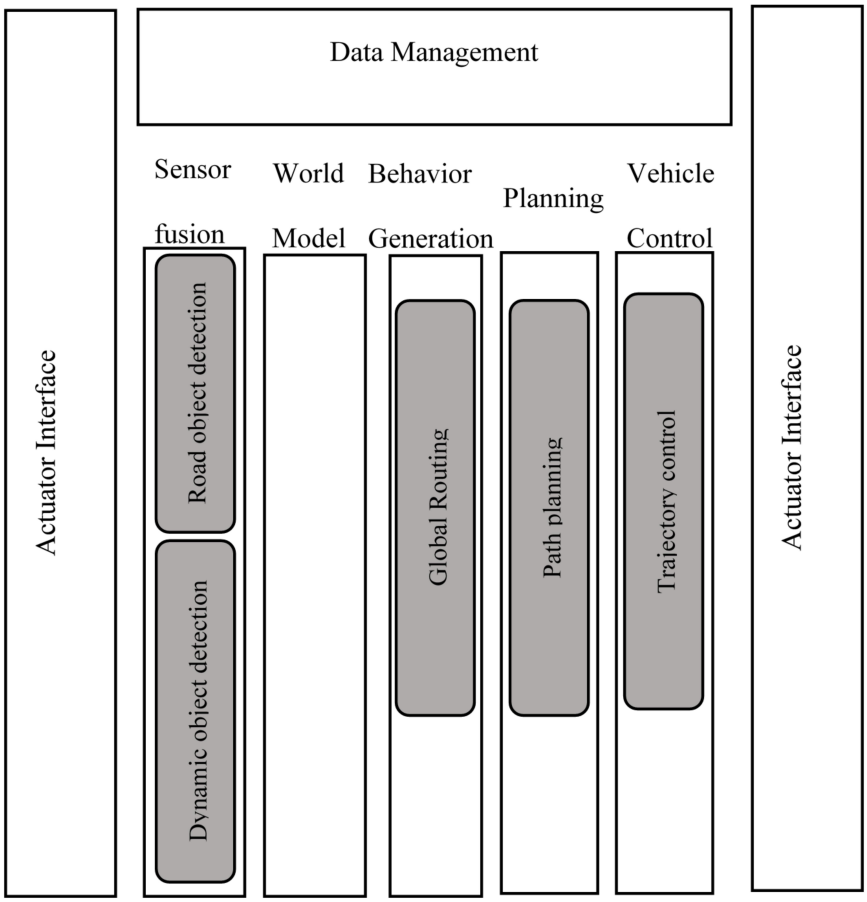

Figure 5. Selected functional components for the case study.

In this case study, we considered six components, $CO = \{CO_1, CO_2, \cdots, CO_6\}$, which are described in Table 4. In the rest of this part, the phases of the proposed framework for this case study are applied.

Table 4. Components' attributes.

Component Name	Component Attributes
CO_1	Solution based on the following attributes: • 3-layer Neural Network is used as the main element in the learning process. • The security issue is considered in data management.
CO_2	Solution based on the following attributes: • Deep Neural Network is used in the learning process. The number of layers is more than three. • high computational power is required by this component.
CO_3	Solution based on the following attributes: • Genetic algorithm is used to organize the learning process. • The Privacy issue is considered in the data management.
CO_4	Solution based on the following attributes: • Deep Neural Network is used as a major element in the cognitive process. • Genetic algorithm is fused to the deep neural network to better organize the learning process. • The learning process of this process requires high computational power.
CO_5	Solution based on the following attributes: • Learning Automata theory is used to manage the learning process. A Linear learning algorithm is used in the learning automata. • The security issue is considered in data management. • The computational power required by this solution is ultra-low.
CO_6	Solution based on the following attributes: • An Ant colony is used to organize the learning process. • The privacy issue is considered in data management.

5.1. Component Analysis Phase

In this phase, functions and component attribute sets are extracted. Table 4 represents some information about all components. Moreover, it is assumed that two functions ($F = \{f_1, f_2\}$) should be operated by the components as described below.

- Object detection (f_1): This function is explicitly mentioned in [12] that should be covered by road object detection and dynamic object detection components. It is assumed that the $z_{f_1} = \{CO_1, CO_2, CO_3\}$.
- Function approximation (f_2): This function is implicitly mentioned in [12] that should be covered by some components such as global routing, path planning, and trajectory control components. It is assumed that the $z_{f_2} = \{CO_4, CO_5, CO_6\}$.

5.2. Criteria Extraction and Weighting

In this phase, four criteria, $CR = \{cr_1, cr_2, cr_3, cr_4\}$ = {Energy Consumption, Security, Privacy, Complexity}, are extracted from dark sides of AI which are discussed in Section 4.2.1. The weighting procedure based on AHP for determining the weights of components is explained in the rest of this phase. As mentioned before, creating a hierarchy for the decision is the first step of AHP analysis. The hierarchy proposed for our use case is depicted in Figure 6. Object detection and function approximation are considered as two main goals (functions); and energy consumption, security, privacy, and complexity are considered as decision criteria. Alternatives (components) are described in Table 4. Components 1, 2, and 3 are alternative ways to reach object detection goal, and three other components are utilized as alternatives for the second goal (function approximation).

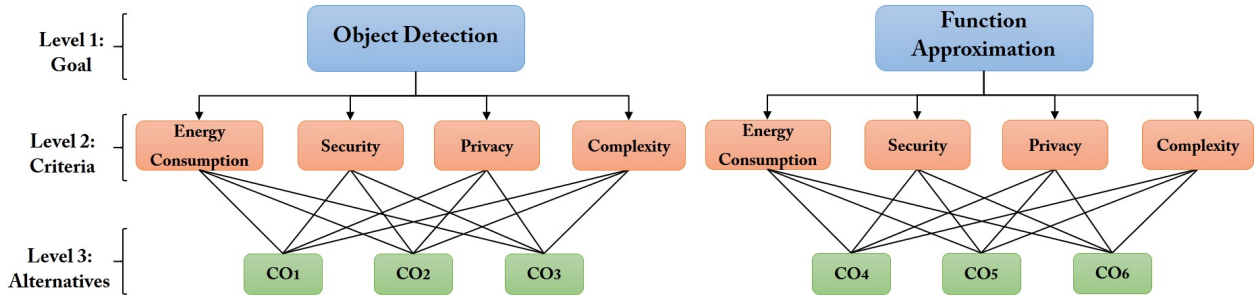

Figure 6. Decision hierarchy for object detection and function approximation.

In order to drive weights for the criteria, a pairwise comparison was created for each goal via comparison matrix as shown in Tables 5 and 6. These matrices were designed in the form of a questionnaire that was answered by 28 AI and software engineering experts. An example of a completed questionnaire for object detection goal is illustrated in Table 7 in which the selected preferences by one of the experts are highlighted. The numerical rating description is considered the same as Table 2. Experts were asked to keep in mind the following proposition while answering the questionnaire: In pairwise comparisons, when comparing two criteria, if the preference is with the item on the left (criterion *i*), one of the left-hand side numerical cells of the table should be marked, and if the preference is with an item on the right (criterion *j*), one of the right-hand side numerical cells of the table should be marked according to the scale mentioned in Table 2. As can be seen in the pairwise comparison matrices (Tables 5 and 6), the numerical values at the bottom of the matrices are inversely proportional to the numerical values at the top of the matrix. The geometric mean of responses was used to combine pairwise comparisons. Weights were then normalized to give a better perspective from criteria.

Table 5. Pairwise comparison matrix of criteria for object detection.

Comparison Matrix (Criteria) for f_1	Energy Consumption	Security	Privacy	Complexity	Geometric Mean	Normalized Weight
Energy Consumption	1	0.137	0.155	5.33	0.58	0.087
Security	7.26	1	5.52	7.41	4.151	0.618
Privacy	6.42	0.181	1	8.08	1.75	0.26
Complexity	0.187	0.134	0.123	1	0.235	0.035

Table 6. Pairwise comparison matrix of criteria for function approximation.

Comparison Matrix (Criteria) for f_2	Energy Consumption	Security	Privacy	Complexity	Geometric Mean	Normalized Weight
Energy Consumption	1	0.177	0.268	8.21	0.79	0.135
Security	5.62	1	4.2	6.02	3.452	0.591
Privacy	3.73	0.238	1	3.46	1.323	0.227
Complexity	0.121	0.166	0.289	1	0.276	0.047

Table 7. An example of a complete questionnaire for object detection.

Criterion i	Comparison Based on Weights										Criterion j
Energy Consumption	9	7	5		3	1	3	5	7	9	Security
Energy Consumption	9	7	5		3	1	3	5	7	9	Privacy
Energy Consumption	9	7	5	4	3	1	3	5	7	9	Complexity
Security	9	7	5		3	1	3	5	7	9	Privacy
Security	9	7	5		3	1	3	5	7	9	Complexity
Privacy	9	7	5		3	1	3	5	7	9	Complexity

The third step is driving relative preferences regarding alternatives by considering each criterion. A questionnaire, which is identical to the criteria questionnaire, was designed and completed by AI and software engineering experts to drive alternatives preferences with respect to each criterion. There were three alternatives and four criteria for each goal in our case study. Hence, four comparison matrices corresponding to the following comparisons were required for each goal. For example, the following comparisons were required for the object detection function:

- By considering the energy consumption criterion: compare CO_1, CO_2, and CO_3 (the results are represented in Table 8).
- By considering the security criterion: Compare CO_1, CO_2, and CO_3 (the results are represented in Table 9).
- By considering the privacy criterion: Compare CO_1, CO_2, and CO_3 (the results are represented in Table 10).
- By considering the complexity criterion: Compare CO_1, CO_2, and CO_3 (the results are represented in Table 11).

The same process took place for the second goal and the results are represented through Tables 12–15. The geometric mean of responses and normalization were used for weight computation. The obtained results are summarized in Table 16.

Table 8. Comparison between suitable components for object detection with respect to energy consumption.

Object Detection-Energy Consumption	CO_1	CO_2	CO_3	Geometric Mean	Normalized Weight
CO_1	1	0.126	0.184	0.285	0.065
CO_2	7.91	1	2.93	2.851	0.653
CO_3	5.42	0.341	1	1.227	0.282

Table 9. Comparison between suitable components for object detection with respect to security.

Object Detection-Security	CO_1	CO_2	CO_3	Geometric Mean	Normalized Weight
CO_1	1	7.83	8.71	4.085	0.8
CO_2	0.127	1	2.13	0.646	0.127
CO_3	0.114	0.469	1	0.376	0.073

Table 10. Comparison between suitable components for object detection with respect to privacy.

Object Detection-Privacy	CO_1	CO_2	CO_3	Geometric Mean	Normalized Weight
CO_1	1	0.495	0.126	0.396	0.08
CO_2	2.02	1	0.124	0.630	0.125
CO_3	7.92	8.01	1	3.988	0.795

Table 11. Comparison between suitable components for object detection with respect to complexity.

Object Detection-Complexity	CO_1	CO_2	CO_3	Geometric Mean	Normalized Weight
CO_1	1	3.14	0.303	0.983	0.215
CO_2	0.318	1	0.343	0.477	0.104
CO_3	3.29	2.91	1	3.123	0.681

Table 12. Comparison between suitable components for function approximation with respect to energy consumption.

Function Approximation-Energy Consumption	CO_4	CO_5	CO_6	Geometric Mean	Normalized Weight
CO_4	1	8.78	7.46	4.030	0.8
CO_5	0.113	1	1.86	0.594	0.118
CO_6	0.134	0.537	1	0.415	0.082

Table 13. Comparison between suitable components for function approximation with respect to security.

Function Approximation-Security	CO_4	CO_5	CO_6	Geometric Mean	Normalized Weight
CO_4	1	0.128	0.252	0.318	0.065
CO_5	7.77	1	6.93	3.776	0.767
CO_6	3.96	0.144	1	0.829	0.168

Table 14. Comparison between suitable components for function approximation with respect to privacy.

Function Approximation-Privacy	CO_4	CO_5	CO_6	Geometric Mean	Normalized Weight
CO_4	1	0.261	0.132	0.325	0.068
CO_5	3.83	1	0.160	0.849	0.178
CO_6	7.56	6.23	1	3.611	0.754

Table 15. Comparison between suitable components for function approximation with respect to complexity.

Function Approximation-Complexity	CO_4	CO_5	CO_6	Geometric Mean	Normalized Weight
CO_4	1	8.43	6.98	3.889	0.783
CO_5	0.118	1	3.18	0.721	0.146
CO_6	0.143	0.314	1	0.355	0.071

Table 16. Summarized relative weight matrix of the alternatives (components).

Summary of Comparisons	Energy Consumption	Security	Privacy	Complexity
CO_1	0.065	0.8	0.08	0.215
CO_2	0.653	0.127	0.125	0.104
CO_3	0.282	0.073	0.795	0.681
CO_4	0.8	0.065	0.068	0.783
CO_5	0.118	0.767	0.178	0.146
CO_6	0.082	0.168	0.754	0.071

The final step is to calculate the final weight of the alternatives. For this purpose, the relative weight matrix of the alternatives (Table 16) must be multiplied by the criteria weight matrix (Tables 5 and 6), which is given below for each function:

Function 1: Object Detection

- $CO_1 : (0.065 \times 0.087) + (0.8 \times 0.618) + (0.08 \times 0.26) + (0.215 \times 0.035) = 0.52838.$
- $CO_2 : (0.653 \times 0.087) + (0.127 \times 0.618) + (0.125 \times 0.26) + (0.104 \times 0.035) = 0.17143.$
- $CO_3 : (0.282 \times 0.087) + (0.073 \times 0.618) + (0.795 \times 0.26) + (0.681 \times 0.035) = 0.30018.$

Function 2: Function Approximation

- $CO_4 : (0.8 \times 0.135) + (0.065 \times 0.591) + (0.068 \times 0.227) + (0.783 \times 0.047) = 0.19865.$
- $CO_5 : (0.118 \times 0.135) + (0.767 \times 0.591) + (0.178 \times 0.227) + (0.146 \times 0.047) = 0.51649.$
- $CO_6 : (0.082 \times 0.135) + (0.168 \times 0.591) + (0.754 \times 0.227) + (0.071 \times 0.047) = 0.28485.$

The value (total weight) of each component is represented in Table 17.

Table 17. Total weights of components.

Component Name	Component Weight
CO_1	$v_1 = 0.52838$
CO_2	$v_2 = 0.17143$
CO_3	$v_3 = 0.30018$
CO_4	$v_4 = 0.19865$
CO_5	$v_5 = 0.51649$
CO_6	$v_6 = 0.28485$

It is worth noting that the CR was under 0.1 for our experimental case study.

5.3. Problem Formulation

In this phase, according to the weights of components, the knapsack problem is customized as follows. We assumed the profit values of all components equal to one. Equation (7) is modified as follows since one of the components should be selected to perform the desired function.

$$\sum_{i=1}^{m} x_{ij} \quad = 1, \; j \; \in 1 \; ..n \tag{13}$$

In addition, parameters m and n are set to 5 and 6, respectively. This means that there are 5 knapsacks.

5.4. Finding Components

In this phase, in order to find the solution to the knapsack problem which formulated in the previous phase, a learning automata-based algorithm is utilized. The reward and penalty parameters of the learning automata are set according to [68]. The final solution set, S, is $\{(K_1, f_1, CO_2), (K_2, f_1, CO_2), (K_3, f_2, CO_4), (K_4, f_2, CO_4), (K_5, f_2, CO_4)\}$ in which CO_2 is selected for functions 1, and CO_4 is selected for function 2. In order to more clarification regarding the findings, some other scenarios are considered in Appendix A.

6. Managerial Implications of the Proposed Framework

In this section, managerial implications of the proposed framework are studied as below:

- When the number of the components increases, component selection considering the dark sides of AI leads to a challenging problem and in some situations lead to NP-hard problem. The proposed framework is the first machine learning algorithm to solve the problem.
- Without this framework, a software package may be easily consisting of components that are not suitable and may also lead to inappropriate software with low accuracy. For example, since in the desired case study, functionality and accuracy were the two most important aspects for desired goals, CO2 and CO4 were selected automatically.
- The proposed mechanism can be used in every software design perspective that invests in the component selection method. Therefore, this method will have many applications including transportation, and healthcare. In addition, the weighting mechanism of the proposed framework may be extended with other multi-criteria decision-making methods such as TOPSIS [71], if there is no correlation between the desired criteria.

7. Conclusions

In this paper, IS design is represented as a CS problem. ISs utilize various AI-based functions to perform the desired functions. A new framework that considers the dark sides

of AI techniques in its proposed solution for ISs' CS problem is proposed in this paper. This framework consists of four phases, namely, component analyzing, extracting criteria and weighting, formulating the problem as multiple knapsacks, and finding components. In the first phase, components' attributes are described and analyzed via experts by considering corresponding functions. In order to extract the criteria, an extensive and comprehensive study on the dark sides of AI techniques is conducted. Dark sides may have intersections among themselves, but each side refers to some unique challenges. Since there is a correlation between the criteria, the AHP method is utilized as a weighting mechanism. After obtaining components' weights, the CS problem is formulated as a multiple knapsack problem, which is solved by using the learning automata algorithm in the last phase, called finding components. The applicability of the proposed framework was investigated through a case study. The autonomous vehicle was considered as an example case study by taking into account its various functional components, including object detection, path planning, and trajectory control. Six components along with four criteria (energy consumption, security, privacy, and complexity) were analyzed and weighted by 28 AI and software engineering experts via the AHP method. The related multiple knapsack problem was solved by the learning automata algorithm assigning an appropriate component to the desired function. It should be noted that a wide range of problems related to technical aspects of dark sides was covered in this paper; and the dark sides of AI may affect other non-technical issues, such as political, social, and financial fields that can be considered as future work.

Author Contributions: Conceptualization, M.R.J. and A.M.S.; data curation, M.R.J. and A.M.S.; methodology, M.R.J. and A.M.S.; software, M.R.J. and A.M.S.; validation, M.S. and A.M.S.; formal analysis, M.R.J.; investigation, M.R.J. and M.S.; resources, M.R.J. and A.M.S.; writing—original draft preparation, M.R.J. and A.M.S.; writing—review and editing, M.S.; visualization, M.R.J. and M.S.; supervision, M.S.; project administration, M.R.J. and A.M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The proposed method invests in questionnaires to calculate weights and then find a specific configuration for the software. In this section, we considered three more scenarios in Table A1 to justify the results.

Table A1. More scenarios.

Scenario	Description	Solution Set
Scenario 1 (K_1, K_2, K_3 requires f_1, K_4, K_5, K_6 requires f_2)	M = 6	$S = \{(K_1, f_1, CO_2), (K_2, f_1, CO_2), (K_3, f_1, CO_2), (K_4, f_2, CO_4), (K_5, f_2, CO_4), (K_6, f_2, CO_4)\}$
Scenario 2 (K_1, K_3, K_5 requires f_1, K_2, K_4, K_6, k_7 requires f_2)	M = 7	$S = \{(K_1, f_1, CO_2), (K_2, f_2, CO_4), (K_3, f_1, CO_2), (K_4, f_2, CO_4), (K_5, f_1, CO_2), (K_6, f_2, CO_4), (K_7, f_2, CO_4)\}$
Scenario 3 (K_1, K_3, K_5, K_7 requires f_1, K_2, K_4, K_6, K_8 requires f_2)	M = 8	$S = \{(K_1, f_1, CO_2), (K_2, f_2, CO_4), (K_3, f_1, CO_2), (K_4, f_2, CO_4), (K_5, f_1, CO_2), (K_6, f_2, CO_4), (K_7, f_1, CO_2), (K_8, f_2, CO_4)\}$

References

1. Szyperski, C.; Gruntz, D.; Murer, S. *Component Software: Beyond Object-Oriented Programming*; Pearson Education: Harlow, UK, 2002.
2. Albert, C.; Brownsword, L.; Bentley, D.; Bono, T.; Morris, E.; Pruitt, D. Evolutionary process for integrating COTS-based systems (EPIC). In *CMU/SEI Technical Report CMU/SEI-2002-TR-005*; Software Engineering Institute: Pittsburgh, PA, USA, 2002.

3. Comella-Dorda, S.; Dean, J.C.; Morris, E.; Oberndorf, P. A process for COTS software product evaluation. In *International Conference on COTS-Based Software Systems*; Springer: Berlin, Germany, 2002; pp. 86–96.
4. Mancebo, E.; Andrews, A. A strategy for selecting multiple components. In Proceedings of the 2005 ACM Symposium on Applied Computing, Santa Fe, NM, USA, 13–17 March 2005; pp. 1505–1510.
5. Haghpanah, N.; Moaven, S.; Habibi, J.; Kargar, M.; Yeganeh, S.H. Approximation algorithms for software component selection problem. In Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC'07), Aichi, Japan, 4–7 December 2007; pp. 159–166.
6. Ballurio, K.; Scalzo, B.; Rose, L. Risk reduction in cots software selection with basis. In *International Conference on COTS-Based Software Systems*; Springer: Berlin, Germany, 2002; pp. 31–43.
7. Virmajoki, J. Detecting Code Smells Using Artificial Intelligence: A Prototype. Master's Thesis, LUT University, Lappeenranta, Finland, 2020.
8. Jiang, F.; Jiang, Y.; Zhi, H.; Dong, Y.; Li, H.; Ma, S.; Wang, Y.; Dong, Q.; Shen, H.; Wang, Y. Artificial intelligence in healthcare: Past, present and future. *Stroke Vasc. Neurol.* **2017**, *2*, 230–243. [CrossRef]
9. Culkin, R.; Das, S.R. Machine learning in finance: The case of deep learning for option pricing. *J. Investig. Manag.* **2017**, *15*, 92–100.
10. Li, J.J.; Bonn, M.A.; Ye, B.H. Hotel employee's artificial intelligence and robotics awareness and its impact on turnover intention: The moderating roles of perceived organizational support and competitive psychological climate. *Tour. Manag.* **2019**, *73*, 172–181. [CrossRef]
11. Gursoy, D.; Chi, O.H.; Lu, L.; Nunkoo, R. Consumers acceptance of artificially intelligent (AI) device use in service delivery. *Int. J. Inf. Manag.* **2019**, *49*, 157–169. [CrossRef]
12. Serban, A.C.; Poll, E.; Visser, J. A standard driven software architecture for fully autonomous vehicles. In Proceedings of the 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), Seattle, WA, USA, 30 April–4 May 2018; pp. 120–127.
13. Kim, H.; Pyeon, H.; Park, J.S.; Hwang, J.Y.; Lim, S. Autonomous Vehicle Fuel Economy Optimization with Deep Reinforcement Learning. *Electronics* **2020**, *9*, 1911. [CrossRef]
14. Syam, N.; Sharma, A. Waiting for a sales renaissance in the fourth industrial revolution: Machine learning and artificial intelligence in sales research and practice. *Ind. Mark. Manag.* **2018**, *69*, 135–146. [CrossRef]
15. Vithayathil Varghese, N.; Mahmoud, Q.H. A survey of multi-task deep reinforcement learning. *Electronics* **2020**, *9*, 1363. [CrossRef]
16. Mora, C.; Rollins, R.L.; Taladay, K.; Kantar, M.B.; Chock, M.K.; Shimada, M.; Franklin, E.C. Bitcoin emissions alone could push global warming above 2 C. *Nat. Clim. Chang.* **2018**, *8*, 931–933. [CrossRef]
17. Strubell, E.; Ganesh, A.; McCallum, A. Energy and policy considerations for deep learning in NLP. *arXiv* **2019**, arXiv:1906.02243.
18. Jaseena, K.; David, J.M. Issues, challenges, and solutions: Big data mining. *CS IT-CSCP* **2014**, *4*, 131–140.
19. Sivarajah, U.; Kamal, M.M.; Irani, Z.; Weerakkody, V. Critical analysis of Big Data challenges and analytical methods. *J. Bus. Res.* **2017**, *70*, 263–286. [CrossRef]
20. Huang, L.; Joseph, A.D.; Nelson, B.; Rubinstein, B.I.; Tygar, J.D. Adversarial machine learning. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, Chicago, IL, USA, 21 October 2011; pp. 43–58.
21. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19. [CrossRef]
22. Bellamy, R.K.; Dey, K.; Hind, M.; Hoffman, S.C.; Houde, S.; Kannan, K.; Lohia, P.; Martino, J.; Mehta, S.; Mojsilović, A.; et al. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM J. Res. Dev.* **2019**, *63*, 4:1–4:15. [CrossRef]
23. Guo, A.; Kamar, E.; Vaughan, J.W.; Wallach, H.; Morris, M.R. Toward Fairness in AI for People with Disabilities: A Research Roadmap. *arXiv* **2019**, arXiv:1907.02227.
24. Haddadin, S. *Towards Safe Robots: Approaching Asimov's 1st Law*; Springer: Berlin, Germany, 2013; Volume 90.
25. Murphy, R.; Woods, D.D. Beyond Asimov: The three laws of responsible robotics. *IEEE Intell. Syst.* **2009**, *24*, 14–20. [CrossRef]
26. Russell, S.; Dewey, D.; Tegmark, M. Research priorities for robust and beneficial artificial intelligence. *AI Mag.* **2015**, *36*, 105–114. [CrossRef]
27. Dawson, J. *Logical Dilemmas: The Life and Work of Kurt Gödel*; AK Peters/CRC Press: Wellesley, MA, USA, 1996.
28. Hofstadter, D.R. *I Am a Strange Loop*; Basic Books: New York, NY, USA, 2007.
29. Došilović, F.K.; Brčić, M.; Hlupić, N. Explainable artificial intelligence: A survey. In Proceedings of the 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 21–25 May 2018; pp. 0210–0215.
30. Schmid, U.; Ragni, M.; Gonzalez, C.; Funke, J. The challenge of complexity for cognitive systems. *Cogn. Syst. Res.* **2011**, *12*, 211–218. [CrossRef]
31. Neri, E.; Coppola, F.; Miele, V.; Bibbolino, C.; Grassi, R. Artificial intelligence: Who is responsible for the diagnosis? *La Radiol. Med.* **2020**, *125*. [CrossRef] [PubMed]
32. Tiwari, A.; Jaga, O.P. Component selection for an electric vehicle: A review. In Proceedings of the 2017 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC), Melmaruvathur, India, 22–23 March 2017; pp. 492–499.

33. Lehmann, M.; Mai, T.L.; Wollschlaeger, B.; Kabitzsch, K. Reducing component selection complexity by component aggregation using design criteria. In Proceedings of the IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23–26 October 2016; pp. 7095–7100.
34. Patel, A.M.; Singal, S.K. Optimal component selection of integrated renewable energy system for power generation in stand-alone applications. *Energy* **2019**, *175*, 481–504. [CrossRef]
35. Salehi, H.; Das, S.; Biswas, S.; Burgueño, R. Data mining methodology employing artificial intelligence and a probabilistic approach for energy-efficient structural health monitoring with noisy and delayed signals. *Expert Syst. Appl.* **2019**, *135*, 259–272. [CrossRef]
36. Hao, M.; Li, H.; Luo, X.; Xu, G.; Yang, H.; Liu, S. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. Ind. Inform.* **2019**, *16*, 6532–6542. [CrossRef]
37. Tsang, Y.P.; Wong, W.C.; Huang, G.; Wu, C.H.; Kuo, Y.; Choy, K.L. A Fuzzy-Based Product Life Cycle Prediction for Sustainable Development in the Electric Vehicle Industry. *Energies* **2020**, *13*, 3918. [CrossRef]
38. Kuwabara, K. Utilizing Dynamic Programming to Aid in the Hybrid Electric Vehicle (HEV) Component Selection Process to Minimize the Vehicle's Fuel Consumption. Ph.D. Thesis, The Ohio State University, Columbus, OH, USA, 2019.
39. Choi, H.D.; Ahn, C.K.; Lim, M.T.; Song, M.K. Dynamic output-feedback $H\infty$ control for active half-vehicle suspension systems with time-varying input delay. *Int. J. Control Autom. Syst.* **2016**, *14*, 59–68. [CrossRef]
40. Wang, R.; Jing, H.; Wang, J.; Chadli, M.; Chen, N. Robust output-feedback based vehicle lateral motion control considering network-induced delay and tire force saturation. *Neurocomputing* **2016**, *214*, 409–419. [CrossRef]
41. Boukens, M.; Boukabou, A.; Chadli, M. Robust adaptive neural network-based trajectory tracking control approach for nonholonomic electrically driven mobile robots. *Robot. Auton. Syst.* **2017**, *92*, 30–40. [CrossRef]
42. Fox, M.R.; Brogan, D.C.; Reynolds, P.F. Approximating component selection. In Proceedings of the 2004 Winter Simulation Conference, Washington, DC, USA, 5–8 December 2004; Volume 1.
43. Petty, M.D.; Weisel, E.W.; Mielke, R.R. Computational complexity of selecting components for composition. In *Fall 2003 Simulation Interoperability Workshop*; Citeseer: University Park, PA, USA, 2003; pp. 14–19.
44. Carlson, S.E. Genetic algorithm attributes for component selection. *Res. Eng. Des.* **1996**, *8*, 33–51. [CrossRef]
45. Baker, P.; Harman, M.; Steinhofel, K.; Skaliotis, A. Search based approaches to component selection and prioritization for the next release problem. In Proceedings of the 2006 22nd IEEE International Conference on Software Maintenance, Philadelphia, PA, USA, 24–27 September 2006; pp. 176–185.
46. Bartholet, R.G.; Brogan, D.C.; Reynolds, P. The computational complexity of component selection in simulation reuse. In Proceedings of the Winter Simulation Conference, Orlando, FL, USA, 4 December 2005; p. 10.
47. Vescan, A.; Şerban, C. A fuzzy-based approach for the multilevel component selection problem. In *International Conference on Hybrid Artificial Intelligence Systems*; Springer: Berlin, Germany, 2016; pp. 463–474.
48. Ernst, N.; Kazman, R.; Bianco, P. Component Comparison, Evaluation, and Selection: A Continuous Approach. In Proceedings of the 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, 25–26 March 2019; pp. 87–90.
49. Kaur, J.; Tomar, P. Clustering based architecture for software component selection. *Int. J. Mod. Educ. Comput. Sci.* **2018**, *11*, 33. [CrossRef]
50. Chatzipetrou, P.; Papatheocharous, E.; Wnuk, K.; Borg, M.; Alégroth, E.; Gorschek, T. Component attributes and their importance in decisions and component selection. *Softw. Qual. J.* **2019**, 1–27. [CrossRef]
51. Rezvanian, A.; Saghiri, A.M.; Vahidipour, S.M.; Esnaashari, M.; Meybodi, M.R. *Recent Advances in Learning Automata*; Springer: Berlin, Germany, 2018; Volume 754.
52. Thathachar, M.A.; Sastry, P.S. Varieties of learning automata: An overview. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2002**, *32*, 711–722. [CrossRef]
53. Saghiri, A.M.; Meybodi, M.R. An approach for designing cognitive engines in cognitive peer-to-peer networks. *J. Netw. Comput. Appl.* **2016**, *70*, 17–40. [CrossRef]
54. Saghiri, A.M.; Meybodi, M.R. An adaptive super-peer selection algorithm considering peers capacity utilizing asynchronous dynamic cellular learning automata. *Appl. Intell.* **2018**, *48*, 271–299. [CrossRef]
55. Misra, S.; Oommen, B.J. Dynamic algorithms for the shortest path routing problem: learning automata-based solutions. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2005**, *35*, 1179–1192. [CrossRef] [PubMed]
56. Agrawal, D.; Das, S.; El Abbadi, A. Big data and cloud computing: Current state and future opportunities. In Proceedings of the 14th International Conference on Extending Database Technology, Uppsala, Sweden, 21–24 March 2011; pp. 530–533.
57. Baig, M.I.; Shuib, L.; Yadegaridehkordi, E. Big Data Tools: Advantages and Disadvantages. *J. Soft Comput. Decis. Support Syst.* **2019**, *6*, 14–20.
58. Greenstadt, R.; Beal, J. Cognitive security for personal devices. In Proceedings of the 1st ACM Workshop on Workshop on AISec, Alexandria, VA, USA, 27 October 2008; pp. 27–30.
59. Kinsner, W. Towards cognitive security systems. In Proceedings of the 2012 IEEE 11th International Conference on Cognitive Informatics and Cognitive Computing, Kyoto, Japan, 22–24 August 2012; p. 539.
60. Varshney, K.R. Engineering safety in machine learning. In Proceedings of the 2016 Information Theory and Applications Workshop (ITA), La Jolla, CA, USA, 31 January–5 February 2016; pp. 1–5.

61. Yampolskiy, R.V. Unpredictability of AI. *arXiv* **2019**, arXiv:1905.13053.
62. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
63. Holzinger, A. From machine learning to explainable AI. In Proceedings of the 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA), Kosice, Slovakia, 23–25 August 2018; pp. 55–66.
64. Grieves, M.; Vickers, J. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary Perspectives on Complex Systems*; Springer: Berlin, Germany, 2017; pp. 85–113.
65. Kim, C.; Kim, Y.; Yi, H. Fuzzy Analytic Hierarchy Process-Based Mobile Robot Path Planning. *Electronics* **2020**, *9*, 290. [CrossRef]
66. Algarín, C.R. An analytic hierarchy process based approach for evaluating renewable energy sources. *Int. J. Energy Econ. Policy* **2017**, *7*, 38–47.
67. Mu, E.; Pereyra-Rojas, M. Understanding the analytic hierarchy process. In *Practical Decision Making*; Springer: Berlin, Germany, 2017; pp. 7–22.
68. Noferesti, S.; Meybodi, M.R. Solving Multidimensional Knapsack Problem using Learning Automata. In Proceedings of the 13th Annual CSI Computer Conference of Iran, Kish Island, Iran, 9–11 March 2008; pp. 7–11.
69. Mamat, N.J.Z.; Daniel, J.K. Statistical analyses on time complexity and rank consistency between singular value decomposition and the duality approach in AHP: A case study of faculty member selection. *Math. Comput. Model.* **2007**, *46*, 1099–1106. [CrossRef]
70. Boulkaibet, I.; Belarbi, K.; Bououden, S.; Chadli, M.; Marwala, T. An adaptive fuzzy predictive control of nonlinear processes based on Multi-Kernel least squares support vector regression. *Appl. Soft Comput.* **2018**, *73*, 572–590. [CrossRef]
71. Behzadian, M.; Otaghsara, S.K.; Yazdani, M.; Ignatius, J. A state-of the-art survey of TOPSIS applications. *Expert Syst. Appl.* **2012**, *39*, 13051–13069. [CrossRef]