future internet

MDPI

Article

AOSR 2.0: A Novel Approach and Thorough Validation of an Agent-Oriented Storage and Retrieval WMS Planner for SMEs, under Industry 4.0

Fareed Ud Din [1,*,†], David Paul [1,†], Joe Ryan [2,†], Frans Henskens [2,†] and Mark Wallis [2,†]

[1] The School of Science and Technology, The Faculty of Science, Agriculture, Business and Law, The University of New England, Armidale, NSW 2350, Australia; dpaul4@une.edu.au
[2] The School of Electrical Engineering and Computing, The College of Engineering, Science and Environment, The University of Newcastle, Callaghan, NSW 2308, Australia; joe.ryan@newcastle.edu.au (J.R.); frans.henskens@newcastle.edu.au (F.H.); mark.wallis@newcastle.edu.au (M.W.)
* Correspondence: fuddin@une.edu.au
† These authors contributed equally to this work.

Abstract: The Fourth Industrial Revolution (Industry 4.0), with the help of cyber-physical systems (CPS), the Internet of Things (IoT), and Artificial Intelligence (AI), is transforming the way industrial setups are designed. Recent literature has provided insight about large firms gaining benefits from Industry 4.0, but many of these benefits do not translate to SMEs. The agent-oriented smart factory (AOSF) framework provides a solution to help bridge the gap between Industry 4.0 frameworks and SME-oriented setups by providing a general and high-level supply chain (SC) framework and an associated agent-oriented storage and retrieval (AOSR)-based warehouse management strategy. This paper presents the extended heuristics of the AOSR algorithm and details how it improves the performance efficiency in an SME-oriented warehouse. A detailed discussion on the thorough validation via scenario-based experimentation and test cases explain how AOSR yielded 60–148% improved performance metrics in certain key areas of a warehouse.

Keywords: agent-oriented smart factory (AOSF); agent-oriented storage and retrieval system (AOSR); cyber-physical systems (CPS); warehouse management system (WMS); small-to-medium-sized enterprises (SME)

Citation: Ud Din, F.; Paul, D.; Ryan, J.; Henskens F.; Wallis, M. AOSR 2.0: A Novel Approach and Thorough Validation of anAgent-Oriented Storage and Retrieval WMS Planner for SMEs, under Industry 4.0. *Future Internet* **2021**, *13*, 155. https://doi.org/10.3390/fi13060155

Academic Editor: Maarit Tihinen

Received: 17 May 2021
Accepted: 4 June 2021
Published: 15 June 2021

1. Introduction

A significant proportion of the world's economy is based on the manufacturing industry [1]. Industrial setups have been evolving ever since their inception. This continuous growth is supported by incorporating process integration, mechanisation of operations and customised procedural manufacturing [2]. The industrial world is now moving towards virtualisation and seamless operations with the help of artificial intelligence [3]. Extensive research and development have provided the manufacturing industry with high-tech solutions to speed up the process of production and delivery of end-products to customers by utilising the concepts of distributed artificial intelligence [4], Internet of Things (IoT) [5], Big Data [6], multi-agent systems (MAS) [7], cloud computing [8], and Industry 4.0 [9]. The initiative of Industry 4.0 recommends IoT-enabled, sustainable, Big Data-driven decision-making processes and digitized mass production within manufacturing systems [10] by utilising advanced infrastructural transformation and incorporating smart machines within the supply chain (SC), having nano- or micro-chips installed in them [11]. In order to build such a structure, high-performance computing devices are required, which ultimately increase the infrastructural and operational cost. Although large setups can afford such solutions, small-to-medium-sized enterprises (SME), which are mostly centrally controlled and mostly not compatible with such advanced systems [12], may lag behind [13].

For SC operations, warehouses serve as the real backbone for maintaining the whole value chain [14]. Handling warehouse operations efficiently is not possible without a warehouse management system (WMS). A WMS is a software application that supports day-to-day operations in a warehouse. Extensive research is conducted to provide complete autonomous warehousing systems, e.g., the solutions with robo-machines and auto-conveyor belts, such as the flow shop algorithm [15] and EMBBO [16] or niche warehousing problems such as the pickup and delivery problem with cross-docking (PDPCD) [17]. Even though the idea of Industry 4.0 is transforming the manufacturing industry via state-of-the-art decision-making algorithms, automated production systems, and Big Data-driven innovation [18,19], recent research claims that Industry 4.0 standards cannot be purely mapped to SMEs [12,13]. The concept of Logistics 4.0 [20] is also a futuristic progression in this domain, which recommends incorporating optimised routing, reduced storage requirements, and autonomous robots with tracking and decision systems for optimised inventory control and information exchange; however, Industry 4.0 is still at its very beginning, and any future impacts on logistics management are rather uncertain [21]. SMEs are still facing warehousing issues, such as wandering items or picking lists [22], inaccurate stock values at runtime [23], unmanaged receiving and expedition areas [24], unmanaged storage capacity, and inappropriate retrieval scheduling [4].

In order to provide an overarching solution to help bring the SMEs many of the benefits of Industry 4.0, the agent-oriented smart factory (AOSF) framework [25] provides a comprehensive and high-level SC architecture for SMEs under the umbrella of Industry 4.0, and the agent-oriented storage and retrieval (AOSR) system [26] presents a complementary addition to the novel approach of the AOSF framework, to provide a low-cost, semi-autonomous warehouse system. The episodic series of this broad contribution includes: a high-level SC architecture of the AOSF framework [25]; its problem and domain definition, which provided the baseline model for the agent-oriented storage and retrieval (AOSR) system [27]; the AOSF's recommended *six-feature strategy* and the general workflow of the AOSR system [26]; the validation of the AOSF framework in comparison with a linear supply chain [28]; the validation of the time efficiency of the AOSR's algorithmic strategy [29]; and the conceptualisation of the AOSF as a customised CPS [30].

The main contribution of this paper is the definition of the extended heuristics of AOSR 2.0. This paper also demonstrates that AOSR improves existing warehouse algorithms. This is performed via scenario-based test cases, in comparison with the standard linear SC-based standard warehouse management system (WMS) strategies. The remainder of this paper is structured as follows. Section 2 describes the methodology used in this paper. This includes a complete overview of the AOSR 2.0 algorithmic heuristics, as well as a description of the methods and data used to compare our new algorithm with existing strategies. This includes a detailed discussion of different standard warehousing strategies for product placement or retrieval, e.g., zone logic, FIFO, and pick from or put to the fewest logic [31]. Section 3 includes the results of scenario-based experiments applied to the AOSR 2.0 Java Agent Development Environment (JADE) [32] prototype, in comparison with standard WMS strategies. The hybrid logic-based strategy of AOSR 2.0 heuristics provides a thorough mechanism to utilise additional logic schemes in suitable scenarios, such as "pick from/put to the nearest logic", on top of the aforementioned schemes, which helps reduce the overall activity-time and improve performance efficiency within the warehouse. Section 4 concludes the paper, suggesting that AOSR 2.0's superior performance demonstrates that it could be used as a drop-in replacement for existing algorithms to improve warehouse performance, and describes future research potentials in this area.

2. Methodology

We previously presented the general AOSR algorithm [26], which provides a simple yet comprehensive solution to the baseline issues of a standard distribution warehouse, especially for scheduling products via its hybrid slotting and re-slotting strategy. The focus of the AOSR strategy is to minimise the main reasons that cause disruptions in general

warehouse activities, such as by trying to reduce the number of the products in certain key areas of a warehouse, e.g., receiving and expedition areas (RA/EA), and maximise the number of products stored in the defined racks [33]. Although the AOSR system achieves the main goal to overcome the aforementioned general warehouse problems, to equip it with fortified sensing ability to detect the percepts from the environment and the defined states of the system (as detailed in the algorithmic heuristics in the later section), some advanced concepts have been employed (e.g., *Belief Base* and *Knowledge Base*, described in Section 2.3), which help in building a dynamic product placement plan, by utilising the hybrid logic-based strategy. Intelligent wireless sensor networks make it possible to obtain an ever-increasing amount of data, which must be analysed efficiently and effectively to support increasingly complex systems' decision-making and management [34,35]. The capability to sense from the environment and predict the upcoming space congestion on the shop floor makes AOSR 2.0 more proactive in nature, which helps in managing the warehouse capacity before reaching a bottleneck situation.

2.1. Objective

The main aim of this paper is to validate the AOSR 2.0 strategy against other standard warehousing strategies using multiple test cases. In order to perform a thorough validation, the test scenarios are subdivided through two major validating parameters: performance efficiency (discussed as part of this article) and time efficiency (discussed in our other work [29]).

The performance efficiency of the AOSR strategy is tested based on three KPIs, in three different types of scenarios for the number of products:

- stored in Racks;
- stuck at a RA; and
- placed in a EA.

In order to provide a clear comparison of the AOSR 2.0 strategy with the other standard approaches, the experiments are performed by combining the standard zoning logic with two widely acceptable warehousing logics: (i) zoning logic with first in first out (FIFO) logic [36], which picks and puts the products based on order of arrival; and (ii) zoning with fewest logic [31], which picks and places products with a preference to pick products from the most empty rack first or to put the products in the fullest rack in which they fit first to maximise available space.

2.2. Data

As detailed in our previous work [26], the AOSR strategy provides a mechanism to handle different classifications of products. It recommends a general structure of different warehouse zones to match the SKU requirements. The racks in each zone can be further divided into different levels. The number of racks and levels are flexible and can be configured initially before launching the setup. As a constraint for experimental purposes in this paper, all the AOSR racks are divided into three levels with each level containing space for five SKUs, yielding a total of 15 SKUs in a rack. This implies that, for a minimal setup, there is storage capacity for around 2000 products (distributed as defined in the dataset used [26]) with the flexibility to support up to 20,000 products to be scheduled in a single day in a larger setup. In order to justify the need and comprehensiveness of data, the industrial dataset utilised to test AOSR 2.0 provides a variety of different classifications of products, e.g., finished goods or raw materials and fast or slow-moving products, in compliance with the test datasets provided by different logistics and warehousing companies e.g., Eurosped [37] and DGI Global [38]. The details of different product categorisations applied to the AOSR 2.0 strategy are discussed in our other work [28], which includes how the products are classified in the datasets. In order to ensure comprehensive experimentation, AOSR 2.0 caters to six different types of SKUs (Each/Box, Box/Case, Case/Pallet, Barrel, Cylinder, Single Pallet) and six different types of product characteristics (hazardous or non-hazardous, fast or slow, and finished good or raw material) with

20 different product categories. Thus, the number of product possibilities that can be catered for can be represented as:

$$n^2 * Categories * Characteristics * SKUs. \quad (1)$$

Then, the possibility of finding a certain product x in n advance shipment notices (ASN) or n advance delivery notices (ADN) that has any particular category out of the 20 possible, with any particular characteristic out of the six possible, and having any particular SKU out of the six possible, can be calculated with the following equation:

$$P_{(x)} = \frac{1}{n^2 * 20 * 6 * 6} \quad (2)$$

In a complex warehouse environment, finding product possibilities can sometimes help to adjust space and product allocation within designated areas. A simplified solution to find a product possibility in a certain warehouse region can help improve search ability and efficiency in a warehouse solution [14]. As AOSR WMS maintains a more generic solution, it becomes easier to apply a different set of requirements as per business requirements.

2.3. Methods: AOSR Heuristics

The foundations of the AOSR algorithm [26] are based on the classical belief desire intention (BDI) agent model architecture [39] in conjunction with the constructs and AOSF's agent categorisation [25], which enables it to support a dynamic environment and remain flexible. As depicted in Figure 1, the architecture of the AOSR 2.0 algorithmic heuristics incorporates concurrent information threads [28] for defined products and racks, as well as their characteristics and categorisations, via an important component called the *Belief Base*, which serves as the main source of truth. Information regarding stock levels and current system states are recorded or supplied via another parallel component called the *Knowledge Base*, which keeps itself updated from the information provided by actuators. The Knowledge Base continually uses actuators to update general information that is taken to be true from the environment and the Belief Base keeps information that is considered a central fact to compare with environmental percepts, but can be revised based on repetitive factual information from the environment. There are three main actuators used in the AOSR 2.0 algorithm: (i) *Search Rack*; (ii) *Placement Generator*; and (iii) *Extract Placement*. These actuators are responsible for sensing the percepts from the environment and updating the Belief and Knowledge Base via the *Knowledge Builder* and *Belief Builder* as detailed in Algorithm 1.

Other supporting sub-functions, such as *extract characteristics* and *Generate CharcID*, which, respectively, find details of the characteristics of a given product or find a matching product based on the given characteristics, are also incorporated within the AOSR 2.0 algorithmic heuristics, which support the role of actuators. The composite architecture of AOSR 2.0 provides the agents with needed properties (e.g., goals or objectives, interaction protocols, timestamped actions, available resources, knowledge and belief sets) and transforms the AOSR 2.0 strategy into a pure agent-oriented solution. The functionality of these components has been implemented in a prototype using JADE [32]. Algorithms 1–5, described later in this section, give details of this implementation. First of all (and as a continuous process), the *Belief Builder* extracts and updates the overall *Belief Base* by utilising the *Thread Reader*, and takes care of any anomalies by utilising the *Check-Exceptions* function. In parallel, *Percept-Builder* senses any upcoming changes for the environment by utilising the *Request-Analyser* function. To process requests, the *Planner Agent* utilises either *SearchRack* or *Placement Generator* and updates the overall plan. Algorithms 1–5 includes the details of the overall flow of the execution.

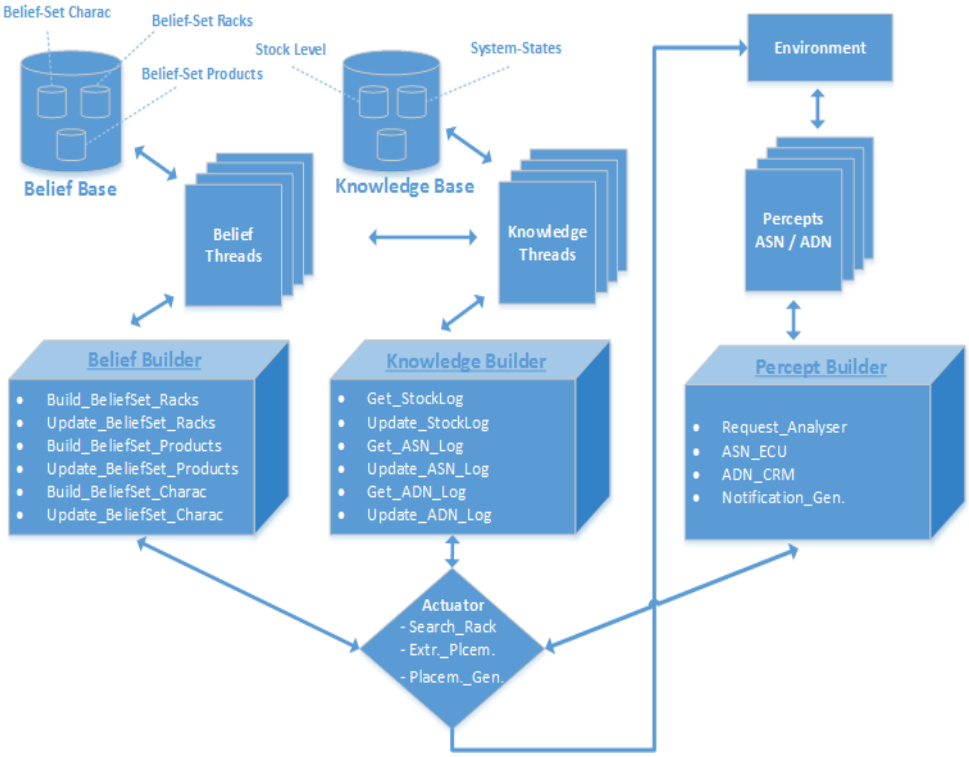

Figure 1. Algorithmic Architecture of AOSR 2.0.

Algorithm 1 Overview of Belief Builder Heuristics.

```
1:  procedure BELIEF-BUILDER
2:      CheckExceptions()
3:      Initiate ThreadReader()
4:  top:
5:      if updateThreadReceived then
6:        UpdateBeliefBase()
7:      end if
8:      BeliefStream[] ← extractBeliefBase()
9:  do:
10:     BeliefThreadProduct[i] ← BeliefStreamProduct[i]
11:     BeliefThreadCharac[i] ← BeliefStreamCharac[i]
12:     BeliefThreadRack[i] ← BeliefStreamRack[i]
13: while (BeliefStream.hasNext())
14:     CheckUpdates
15:     if UpdatesAvailable then
16:         goto top
17:     end if
18: end procedure
```

Flexibility and reconfigurability are key features that make the AOSR 2.0 strategy more adaptable for any implementation environment. All the baseline information sets are stored in a form of Belief Sets (Belief Set Products, Belief Set Characteristics, Belief Set Racks), which build the overall *Belief Base* for the AOSR 2.0. Algorithm 1 presents an overview of how Belief Builder can initiate and update the *Belief Base* for AOSF agents. These Belief Sets can be modified if needed as per any business requirements, which provides volatility in terms of modifying the standard settings based on confidence about the knowledge according to its date, nature, or the sender of the information (i.e., other agents from AOSF environment).

Belief-Builder holds the capability, in certain situations, to check on any upcoming exceptions by utilising the *Check-Exceptions* function. For example, sometimes a request to fetch data may not succeed because of a network failure or an empty record. In such a case, the dataset may hold no record-tuple, which may cause errors when performing operations through actuators. Belief-Builder manages such issues before initiating other components, as highlighted in Algorithm 1. Belief-Builder converts all the information from the *Belief Base* into data threads to read through and thoroughly compare every single data entry and uses *Thread-Reader()* to extract, analyse, and combine each of them into different Belief Streams. For the initial configuration, the Belief Builder builds the baseline beliefs for Products and Racks, as well as their detailed characteristics. Similarly, Knowledge Builder is based on the same strategy to build the pool of knowledge-constructs and maintain a completely updated *Knowledge Base*.

The feature of sensing from the environment ensures AOSR 2.0 is constantly updated, which helps ensure actions are planned in a timely manner. Algorithm 2 represents the heuristics of *Percept-Builder*, which is responsible for pooling percepts from the environment. It builds its beliefs and knowledge from *Belief Builder* and *Knowledge Builder*, respectively. Based on knowledge threads related to products' locations within the warehouse, it builds a comprehensive placement plan (P), which keeps updating whenever a product batch needs to be shipped or delivered.

Algorithm 2 Overview of Percept Builder-ECU Heuristics.

```
1:  procedure REQUEST-ANALYSER-ECU
2:      Initiate BeliefBuilder()
3:      Initiate KnowledgeBuilder()
4:      PlacementPlan[] ← KnowledgeThreadPlan[]
5:      request ← ACLmessageReceiver()
6:      P ← request.requiredProduct
7:      Q ← request.requiredQuantity
8:      C[] ← Extract-Characteristics(P)
9:      if request is from ECU then
10:         AvailableRacks[] ← Search-Rack(P , c[] , Q , Plan[])
11:         if (AvailableRacks[]) then
12:             FewestAvailableRacks[] ← FindFewest(AvailRacks[])
13:             NearestAvailableRack ← FindNearest(FewestAvailableRacks[])
14:             GeneratePlacement(P, Q, NearestAvailableRack)
15:             UpdateBeliefBuilder()
16:             UpdateKnowledgeBuilder()
17:             Notification-Generator(SUCCESS, ECU)
18:         end if
19:         if (AvailableEA()) then
20:             if (CheckReslottingNeed()) then
21:                 p ← ExtractADNlogProduct()
22:                 q ← ExtractADNlogQuanitity()
23:                 GeneratePlacement(p, q, EA)
24:                 GeneratePlacement(p, q, ExtractPlacement(P, Q)
25:                 UpdateBeliefBuilder()
26:                 UpdateKnowledgeBuilder()
27:                 NotificationGenerator(SUCCESS, ECU)
28:             end if
29:         end if
30:         else
31:             NotificationGenerator(FAILURE, ECU)
32:     end if
33:     CheckUpdates
34: end procedure
```

The AOSR-WMS system interacts actively with two main SC entities via several other agents (e.g., supplier side agents, customer side agents, smart device agents, mediator agents, and other user software agents) defined in the AOSF environment [25,30]: (i) enterprise central unit (ECU), which further connects with the supply chain management unit (SCM) to take account for all the suppliers' management; and (ii) customer relationship management unit (CRM), which takes account of all customer interactions. Both units, ECU and CRM, send requests and desires to a warehouse planner agent for any shipment or delivery operations, corresponding to certain product-batches. *Percept-Builder* utilises its method of *Request-Analyser()* to identify two of its variations: requests from ECU side agents; and requests from the CRM side agents; and performs tasks accordingly to the context to ensure it reaches the desires of the requesting agents. The AOSR algorithm completely complies with the FIPA-agent communication language (ACL) protocol [40] to perform negotiation between different agents. All messages between different AOSR components follow ACL constructs. The *ACLmessageReceiver()* function in Algorithm 2 extracts all the subcomponents of a request or desire and identifies the information related to product details, e.g., their SKUs, characteristics, or quantity. The *Extract-Characteristics()* function fetches all the characteristics related to a particular product highlighted in the shipment or delivery details. The AOSR algorithm recommends advance notification of products shipment (ASN) or delivery (ADN), as discussed in AOSR's *6-Feature Strategy* [26]. The set of characteristics included in ASN/ADN helps to find the right match to determine a suitable rack to place the product within the warehouse at an appropriate location.

Algorithm 3 Overview of Actuator-SearchRack Heuristics.

```
1:  procedure ACTUATOR-SEARCHRACK
2:      Initiate Belief-Builder()
3:      Initiate Knowledge-Builder()
4:  foreach P in ASN:
5:      matchCharacteristics(KnowledgeThread(Rack, P))
6:      matchCapacity(KnowledgeThread(Rack, P))
7:      if matched then
8:          AvailableRacks[] ← KnowledgeThread(RackNo)
9:          AvailableRackLevels[] ← KnowledgeThread(RackLevel)
10:     end if
11:     goto loop
12:     if  thenCheckConsolidation(AvailableRacks[], AvailableRackLevels[]))
13:         AvailableRacks[] ← fewest(AvailableRacks[], AvailableRackLevels[]))
14:     else
15:       AvailableRack ← nearest(AvailableRacks[], AvailableRackLevels[]))
16:     end if
17:     return AvailableRack
18: end procedure
```

If a request comes from an ECU side agent, the first step that the AOSR planner agent performs is to find a suitable rack, as explained in Algorithm 3. Other than matching the characteristics of products with those of racks, capacity is one of the main concerns in order to completely store the batch. In contrast to a standard WMS ([4,41]), AOSR 2.0 provides an advanced and deeper approach to assign a rack to a product. It does not just randomly assign a product to an available rack but instead analyses the list of available racks based on capacity and location and then attempts to consolidate a slot within the rack, e.g., by justifying the maximum possible space to completely fill the same rack level (i.e., if a shipment is received with three products and there two of the same product in a rack already, then it will place the new products in the same rack to take it to its full capacity of five products), rather than putting the dispersed products into different locations. Although this is not always achievable because of capacity and quantity mismatch, it first tries and then finds the nearest possible rack, which ultimately reduces

the overall activity-time on the shop floor. If the method of *SearchRack()* cannot find a suitable available rack, only then does it attempt to find an available expedition area (EA) while, in parallel, it checks for any upcoming delivery orders from its *Knowledge Base*. If it perceives that some products need to be delivered within a given time threshold (with the threshold defined by the company, e.g., 3–5 days), it initiates a task to move the existing products from the racks and put the quantity specified in the ADNs into an available EA so it can place products coming through ASNs directly into racks. Then, on the delivery day, it picks the products from EA and ships them. Thus, through this re-slotting mechanism, the warehouse remains more organised and better managed.

If *Request-Analyser* receives a request from CRM side agents, the *Request-Analyser* utilises the *RetrieveLocation()* function, highlighted in Algorithm 4, to build a list of possible locations for the required product and quantity. Similar to the product placement strategy, the location retrieval strategy also ensures it consolidates the racks by finding the minimum possible products to be fetched in order to clear a rack for upcoming products. If it is not possible to consolidate, it identifies the nearest possible location which the product can be picked to reduce the total activity-time. The *RetrieveLocation()* function returns a failure notification, without crashing, only if the required product is not in stock in the desired quantity.

Algorithm 4 Overview of Percept Builder-CRM Heuristics.

```
1:  procedure REQUEST-ANALYSER-CRM
2:      Initiate BeliefBuilder()
3:      Initiate KnowledgeBuilder()
4:      PlacementPlan[] ← KnowledgeThreadPlan[]
5:      request ← ACLmessageReceiver()
6:      P ← request.requiredProduct
7:      Q ← request.requiredQuantity
8:      C[] ← Extract-Characteristics(P)
9:      if request is from CRM then
10:         if (P with Q inStock) then
11:             PossibleLocations[] ← RetrieveLocation(C[], PlacementPlan[])
12:             FewestAvailable[] ← FindFewest(PossibleLocations[])
13:             NearestAvailable ← FindNearest(FewestAvailable[])
14:             ExtractPlacement(P, Q, NearestAvailable)
15:             UpdateBeliefBuilder()
16:             UpdateKnowledgeBuilder()
17:             NotificationGenerator(SUCCESS, CRM)
18:         else
19:             NotificationGenerator(FAILURE, CRM)
20:         end if
21:     end if
22:     CheckUpdates
23: end procedure
```

Every location in an AOSR 2.0 recommended warehouse is given a unique name so that it can be easily identified and managed without any ambiguities. Every location has a placement code that is comprised of rack number, rack level, and characteristics (e.g., finished or raw, fast or slow, and hazardous or not). These configurations can be varied depending on the business need. An overview of *Placement Generator()* is highlighted in Algorithm 5, which first identifies the available space using the heuristics of *SearchRack()* and then extracts all details from the *Knowledge Base*.

Algorithm 5 Overview of Actuator-PlacementGen Heuristics.

```
procedure ACTUATOR-PLACEMENTGEN
    AvailableRack ← SearchRack()
    if matched in KnowledgeBase then
        RackNo ← KnowledgeThreadRackNo(AvailableRack)
        RackLevel ← KnowledgeThreadRackLevel(AvailableRack)
    end if
    Characteristics ← ExtractCharac(BeliefThreads(P.Charac))
    Location ← GenerateLocation(Characteristics, RackLevel, RackNo)
    return Location
end procedure
```

For testing purposes, we have explored the design mechanisms provided by several available tools but the features provided by JADE [32] were determined to be most suitable for the AOSR 2.0 strategy, e.g., JADE provides simple and flexible options for designing multi-agent scenarios with the ability to monitor overall agents' interactions via the sniffer agent module. Constraint-based experimentation has been applied to AOSR 2.0 in JADE to acquire results in comparison with some standard WMS approaches.

3. Results and Discussion

The prototype developed in JADE to validate the AOSR 2.0 strategy utilises a comprehensive dataset to represent large-scale applicability as discussed in Section 2, which includes thorough variation of different product classifications, SKUs, and time-bound situations related to product delivery and shipment. The AOSR 2.0 strategy attempts to more proactively cater to baseline warehousing issues, such as unavailability or inaccuracy of current stock values [23], mismanagement in EAs/RAs [24], mismanaged capacity in defined storage areas [4] and inappropriate product placement or retrieval strategies [42].

The test cases to validate the performance of AOSR 2.0 are segregated in two different states of the system: Initial Static State (*System State (0)* and Regular Dynamic State (*System State (1)*). System State (0) is a preliminary state where there are no products in the warehouse when shipment notices start to arrive for products to be shipped to the warehouse. System State (1) is a normal running-system state where there are already some products stored in the warehouse and both the ASNs and ADNs are being received for products to be shipped and delivered within the same time interval. All these test cases are examined in the following subsections.

3.1. Scenario of Products in Racks

Figure 2 represents the results taken from System State 0. The results reflect the difference between the tested approaches for the applied test dataset for a full routine day. The number of transactions and iterations are divided into hours (represented on the *x*-axis) and the number of products being shipped or delivered to the warehouse (represented on the *y*-axis). The graph reflects that there is no major difference in the two standard warehousing logics, zoning with FIFO logic and zoning with fewest logic. Although the results generated by AOSR 2.0 represent almost the same pattern, the situation is 10–15% better than the other two approaches as it stores more products in the racks than either of the other two strategies, which is considered a high-performance efficiency metric in a warehouse environment [33]. Nonetheless, all three curves move down after the 6th hour because they reach capacity in the constrained environment. AOSR 2.0, in this scenario, follows the same trend as the other techniques (with better performance than the others because it follows the strategy to keep the minimum possible number of products at the product receiving area) as it utilises its hybrid logic and is not offered a situation where its re-slotting strategy can be utilised since no delivery operations are performed. In System State (1) the performance improvement of AOSR 2.0 over the other techniques is more easily noticed, i.e., in Figure 3.

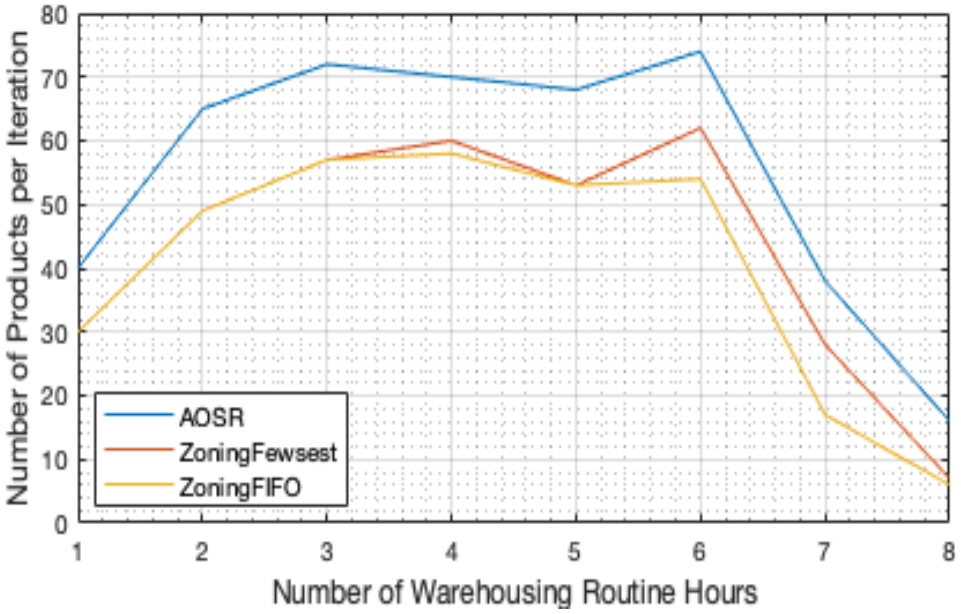

Figure 2. Comparison of Multiple Logics with AOSR 2.0 for Products in Racks in State 0.

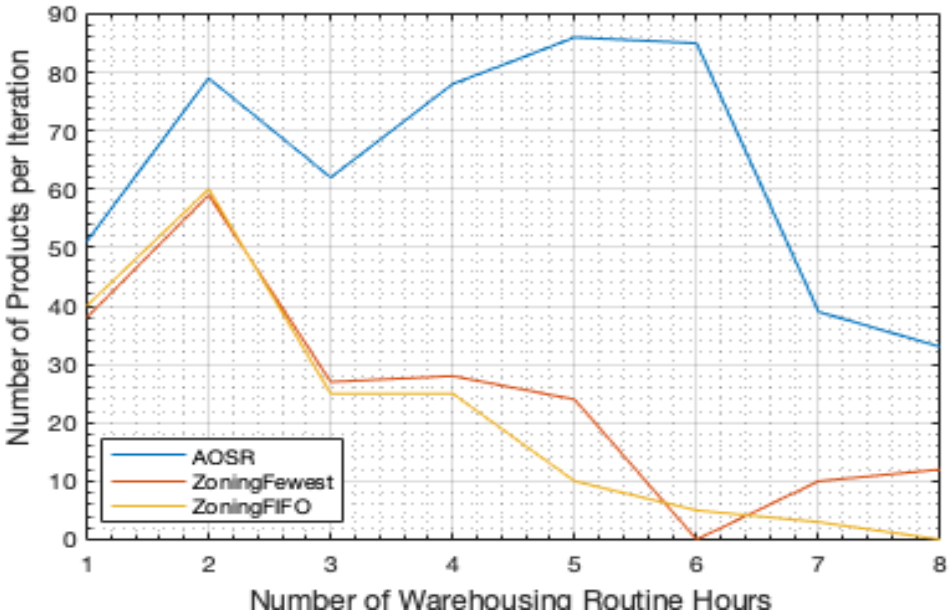

Figure 3. Comparison of Multiple Logics with AOSR 2.0 for Products in Racks State 1.

Figure 3 provides a comparison of the two standard approaches with the hybrid strategy of AOSR 2.0. As can be seen from the graph, for the first 4 hours, both zoning with FIFO logic and zoning with fewest logic have only marginal differences because both of the approaches follow the same pattern of leaving one-quarter of the products in a RA. The main reason for this is because these standard approaches use a manual method of sorting the received products and hence identifying the proper location takes more time [33]. However, the AOSR 2.0 strategy is based on the enterprise integration concepts of the AOSF framework and considers the ASNs or ADNs prior to the arrival of products. Hence, the proactive nature of the AOSR 2.0 strategy already reduces the time taken to identify and place products. After the fourth hour, the performance gap and difference between the AOSR 2.0 strategy and the standard approaches can easily be noticed as AOSR 2.0 utilises its hybrid strategy for product re-slotting to organise space and increase availability for new products. This allows AOSR 2.0 to maintain almost 60% more

products in racks than the other approaches. For the fifth hour, a difference can be seen between the two standard approaches; zoning with fewest logic performed comparatively better than zoning with FIFO logic because it tried to consolidate the space to make more availability for new products to be stored within racks. As the AOSR 2.0 strategy utilises a combination of these approaches, it is more successful and yields better results than the other two individually. A clear performance difference can be seen during the sixth hour, which ultimately reduces for the seventh and eighth hour as the number of total products is reduced in upcoming shipment and delivery notices.

3.2. Scenario of Products in RA

In a standard SC warehouse, a manual method of sorting the received products and identifying the proper location takes almost one-quarter of the total time and operational effort to store products [33]. The case of products in RA is different when utilising the AOSF framework and the *6-Feature Strategy* of AOSR, which recommends business process re-engineering (BPR) based on a proactive approach of sensing ASNs and ADNs, to maintain a minimum possible number of products in the RA by utilising the prior knowledge of upcoming products. The results are shown in Figures 4 and 5 for System State 0 and System State 1, respectively.

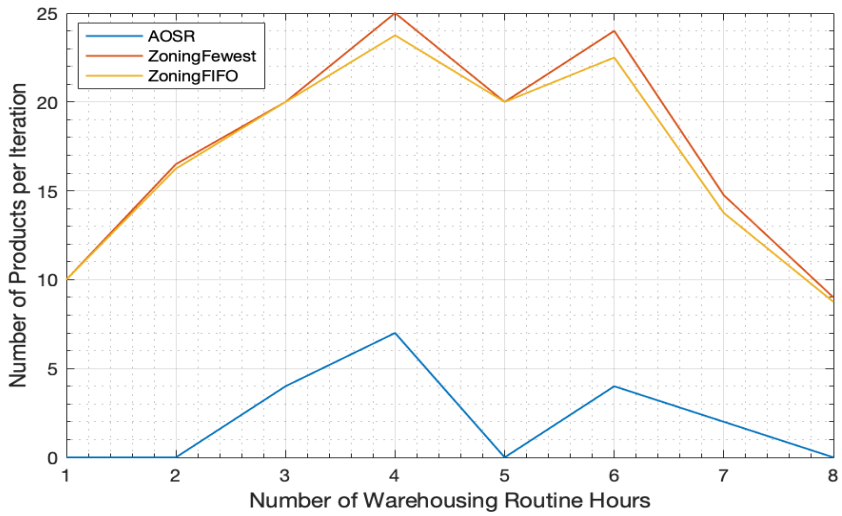

Figure 4. Comparison of Multiple Logics with AOSR 2.0 for Products in RA State 0.

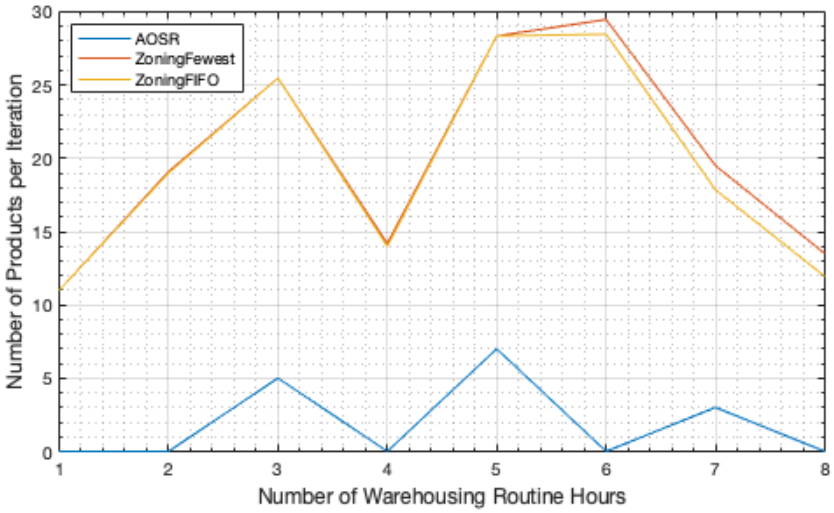

Figure 5. Comparison of Multiple Logics with AOSR 2.0 for Products in RA in State 1.

A difference between the two standard approaches is slightly noticeable in Figure 4, particularly after the third hour, because the zoning with fewest logic has taken more time than the zoning with FIFO logic to sort and identify a proper space for the products, and has, thus, detained more products in the RA. In the case of the AOSR 2.0 strategy, there are very few or no products in the RA because of its proactive approach that allows the AOSR planner agent to make plans before products arrive to ensure the RA is clear for the auto-identification of upcoming products [26].

The difference between the two standard approaches remains unnoticeable in System State 1, as shown in Figure 5, up until the fifth hour. Before then, there is more space available in the warehouse, so both approaches take less time and effort to optimise the available space. Therefore, there is almost the same number of products in RA in both cases. However, when the same products start repeating themselves in upcoming ASNs and ADNs after the fifth hour, the zoning with fewest logic takes more products to RA to identify the available space than the zoning with FIFO logic. In this scenario, the AOSR 2.0 recommended strategy takes the lead and provides up to a 148% decrease in the number of products in RA by incorporating its cognitive and integrative approach to support warehouse activities with its proactive utilisation of its slotting and re-slotting capabilities.

3.2.1. Scenario of Products in EA

For the results acquired in the scenario of products in the EA, there is a very slight difference in the scenarios of System State 0 and System State 1, but they yield considerably different results. The results shown in Figure 6 represent almost no difference in both of the standard logic approaches and the AOSR 2.0 Hybrid Logic as, in System State 0, there are no ADNs, which means products are only received at the warehouse with no product being delivered.

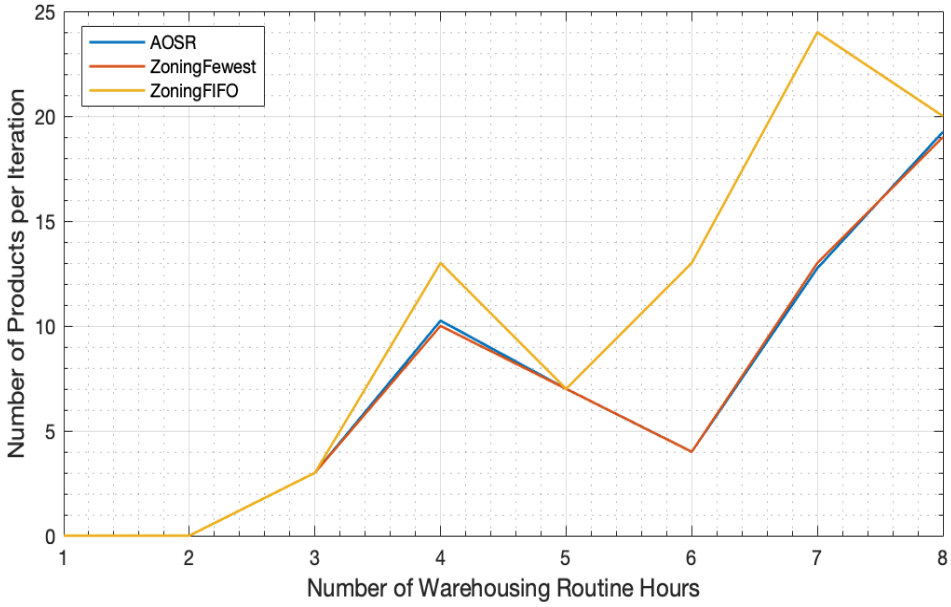

Figure 6. Comparison of Multiple Logics with AOSR 2.0 for Products in EA in State 0.

In this case, the AOSR 2.0 planner algorithm has no opportunity to utilise its re-slotting strategy and shows almost the same pattern as the standard logics. From hours 1 through 3, as there are no products in the racks, all three strategies can easily find the capacity to store products within racks, and the extra products that exceed the total capacity are placed in EA. The difference can be noticed after the third hour, as from the fourth hour onwards, other than at the fifth hour, the zoning with FIFO logic has placed more products in EA because of its failure to incorporate consolidation logic like the zoning with fewest and AOSR 2.0's Hybrid Logic. In the fifth hour, the products appearing in ASNs have

different categories than those already stored in the racks, so the same number of products are placed in EA by each of the three strategies.

The results in System State 1, as represented in Figure 7, clearly demonstrate the performance gap between the approaches. Because both the ASNs and ADNs are being received, AOSR 2.0 can utilise its re-slotting strategy when needed. In the first two hours, there is plenty of space and almost all products are easily stored within the racks, so the number of products in the EA is the same for all of the three strategies. However, after the second hour (and especially in the fourth hour), when the number of products is higher than capacity and when products with the same characteristics appear in upcoming ASNs and ADNs, the difference between AOSR 2.0 and standard strategies is quite visible. The AOSR 2.0 strategy manages to maintain a comparatively lower number of products in the EA throughout the random test scenarios, which can help reduce the issues of wandering or lost items and unmanaged inventory.

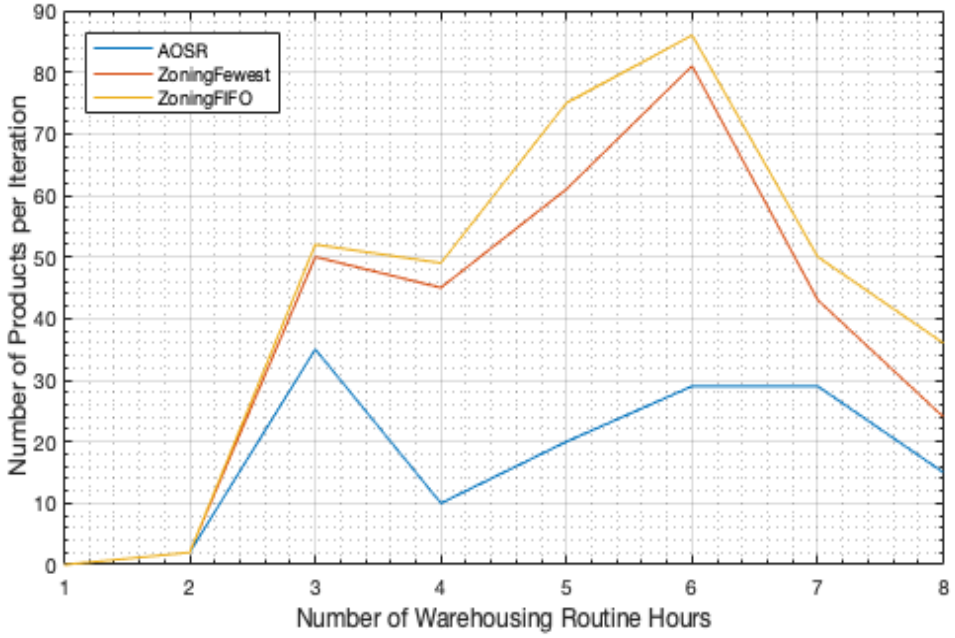

Figure 7. Comparison of Multiple Logics with AOSR 2.0 for Products in EA.

The preference in the *Belief Base* of the planner agent is to place a maximum number of products within the racks. However, when many similar products arrive, so that the total is greater than the maximum capacity of the warehouse for that particular product, the planner agent temporarily places some of the products in the EA. In parallel, it continuously checks with its *Knowledge Base* for any updates about products to be shipped so that it can place the new products into racks rather than the EA and re-slot the products that will soon be needed in the EA from the racks. Then, when the delivery date arrives for the re-slotted products, they can be picked from the EA and space can be cleared for future possibilities. This can be observed during the third and fourth hours in Figure 7, when AOSR 2.0 places products in EA because the number of products in ASNs is much larger than the maximum capacity for that product batch, and it then re-slots the products from racks to the EA and places the newly arriving products into racks so that they do not need to be moved later and inventory can be managed more effectively. Additionally, in the sixth hour, when products with the same characteristics arrive in ASNs, both of the standard logics have placed a very high number of products in the EA, while AOSR 2.0 has placed the products from ADNs in the EA instead allowing the upcoming products to be placed in the racks. This is how AOSR 2.0's re-slotting strategy helps to maintain up to 107% fewer products in EA than the standard approaches. Our previous work [28] also supports the AOSR's performance efficiency overall.

Other than the validation of this system in JADE, we have also validated this system using real data. For this purpose, we implemented the AOSR 2.0 strategy in an industrial simulation tool, Demo3D [43]. Demo3D is a product of RockWell Automation [44], which is a US-based firm providing industrial automation solutions. We created a 3D visualisation of a warehouse with the AOSR 2.0 strategy and achieved comparable results to those discussed in this paper. These results are not included as part of this paper because of privacy concerns of the organisation whose data were used for this implementation.

4. Conclusions and Future Work

This article presented the extended heuristics and performance-based validation of AOSF's associated AOSR strategy in an SME-oriented warehouse environment. The applied test scenarios were discussed in comparison with standard warehousing strategies (Zoning with FIFO logic and zoning with fewest logic), particularly for managing the number of products in key warehouse areas, such as racks, RA, and EA, with different corresponding system states (with or without possible conflicts). The results from different scenarios demonstrate that the AOSR algorithm performs better in comparison with standard WMS strategies, especially in System State (1), which is a normal running state of a warehouse. However, it is impossible for a single solution to be universally applicable, so the presented system also has some limitations. For example, performance has only been tested in a prototype, not in a real-time distributed cloud architecture, where results may vary slightly. Furthermore, this system does not include in-built cloud server security, which is another rich area of research. Although the AOSR strategy can cater to requests coming from smart-devices, connecting manual industrial hardware components to this system may raise some more areas of optimisation.

For large-scale distributed enterprise setups, the AOSR's parent architecture of the AOSF framework can be utilised for inter-enterprise integration as it includes OLAP based systems and server architectures on the cloud layer. The AOSF framework provides a high-level guide for manufacturing industrial management for SMEs, but could be developed further to cater for concerns related to privacy and security. For AOSR, there could be some more dimensions to work upon in the future, such as movement of products within the warehouse shop floor using forklift trucks, utilising collapsible racks or small-scale drones (as some industries offer a high-tech robo-oriented solution such as GrayOrange [45] and Unleashed [46]). These solutions provide nice cutting-edge features but come with an additional infrastructure cost. Similarly, information systems based on cloud-based cognition, Big Data-driven and Deep Learning-assisted process planning and decision-making [47,48] could be implemented for large industrial environments. The moderate level, semi-autonomous, low-cost solution provided by AOSR can also be used for incremental improvements in the future. For example, the system is flexible enough that conveyor belts and picking machines or Big Data support could be added to the system if needed. Additionally, as we have implemented the AOSR strategy in Demo3D [43] in liaison with a local industry which offers consulting services to diverse industrial clients from Australia, South East Asia, and North America, we are in the process of getting more involved in offering this solution to a range of their clients.

Author Contributions: Conceptualization, F.U.D., D.P., F.H., M.W.; Methodology, F.U.D., D.P., F.H., M.W.; Software, F.U.D.; Validation, F.U.D., D.P. and J.R.; Formal analysis, D.P., J.R.; Investigation, F.U.D., D.P., J.R., F.H., M.W.; Resources, F.U.D., D.P.; Data curation, F.U.D.; Writing—original draft preparation, F.U.D.; Writing—review and editing, D.P.; Visualization, F.U.D.; Supervision, D.P., J.R., F.H., M.W.; Project administration, F.U.D., D.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by an Australian Government Research Training Program (RTP) Scholarship under the ID: C3263747 at The University of Newcastle, NSW, Australia.

Data Availability Statement: The details of different classes and categorisation of products in the test cases used in this article are extracted from the online source provided by DGI Global and Eurosped warehousing and logistics companies.

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

References

1. McKinsey, C. The Impact and Future of Manufacturing. 2019. Available online: https://www.mckinsey.com/business-functions/operations/our-insights/the-future-of-manufacturing (accessed on 3 April 2019).
2. Freeman, C.; Louçã, F. *As Time Goes by: From the Industrial Revolutions to the Information Revolution*, 1st ed.; Oxford University Press: Oxford, UK, 2001.
3. Xu, L.D.; Xu, E.L.; Li, L. Industry 4.0: State of the Art and Future Trends. *Int. J. Prod. Res.* **2018**, *56*, 2941–2962. [CrossRef]
4. Lu, W.; Giannikas, V.; McFarlane, D.; Hyde, J. *The Role of Distributed Intelligence in Warehouse Management Systems*; Springer: Cham, Swizerland, 2014; pp. 63–77.
5. Majeed, A.A.; Rupasinghe, T.D. Internet of things (IoT) Embedded Future Supply Chains for Industry 4.0: An Assessment from an ERP-based Fashion Apparel and Footwear Industry. *Int. J. Supply Chain. Manag.* **2017**, *6*, 25–40.
6. Wang, S.; Wan, J.; Zhang, D.; Li, D.; Zhang, C. Towards Smart Factory for Industry 4.0: A Self-Organized Multi-Agent System with Big Data based Feedback and Coordination. *Comput. Net.* **2016**, *101*, 158–168. [CrossRef]
7. Lützenberger, M.; Küster, T.; Masuch, N.; Fähndrich, J. Multi-agent Systems in Practice: When Research Meets Reality. In Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, 9–13 May 2016; pp. 796–805.
8. Jazdi, N. Cyber Physical Systems in the Context of Industry 4.0. In Proceedings of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, 22–24 May 2014; pp. 1–4.
9. Henning, K.; Wolfgang, W.; Johannes, H. Recommendations for Implementing the Strategic Initiative Industrie 4.0. *AcaTech Natl. Acad. Sci. Eng.* **2013**, *1*, 13–57.
10. Riley, C.; Vrbka, J.; Rowland, Z. Internet of Things-enabled Sustainability, Big Data-driven Decision-Making Processes, and Digitized Mass Production in Industry 4.0-based Manufacturing Systems. *J. Self Gov. Manag. Econ* **2021**, *9*, 42–52.
11. Sommer, L. Industrial Revolution Industry 4.0: Are German Manufacturing SMEs the First Victims of this Revolution? *J. Ind. Eng. Manag.* **2015**, *8*, 1512. [CrossRef]
12. Andulkar, M.; Le, D.T.; Berger, U. A multi-case study on Industry 4.0 for SMEs in Brandenburg Germany. In Proceedings of the 51st Hawaii International Conference on System Sciences, Waikoloa Village, HI, USA, 2–6 January 2018; pp. 4544–4553.
13. Müller, J.M.; Buliga, O.; Voigt, K.I. Fortune favors the prepared: How SMEs approach business model innovations in Industry 4.0. *Technol. Forecast. Soc. Chang.* **2018**, *132*, 2–17. [CrossRef]
14. Abu Al-Rejal, H.M.; Abu Doleh, J.D.; Salhieh, L.M.; Udin, Z.M.; Mohtar, S. Barriers of Supply Chain Management Practices in Manufacturing Companies in Republic of Yemen: Pre-War Perspective. *Int. J. Supply Chain. Manag.* **2017**, *6*, 246–251.
15. Centobelli, P.; Converso, G.; Murino, T.; Santillo, L. Flow Shop Scheduling Algorithm to Optimize Warehouse Activities. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 49–66. [CrossRef]
16. Ma, H.; Su, S.; Simon, D.; Fei, M. Ensemble Multi-Objective Biogeography-Based Optimization with Application to Automated Warehouse Scheduling. *Eng. Appl. Artif. Intell.* **2015**, *44*, 79–90. [CrossRef]
17. Santos, F.A.; Mateus, G.R.; Da Cunha, A.S. The pickup and delivery problem with cross-docking. *Comput. Oper. Res.* **2013**, *40*, 1085–1093. [CrossRef]
18. Duft, G.; Durana, P. Artificial Intelligence-based Decision-Making Algorithms, Automated Production Systems, and Big Data-driven Innovation in Sustainable Industry 4.0. *Econ. Manag. Financ. Mark.* **2020**, *15*, 9–18.
19. Hyers, D. Big Data-driven Decision-Making Processes, Industry 4.0 Wireless Networks, and Digitized Mass Production in Cyber-Physical System-based Smart Factories. *Econ. Manag. Financ. Mark.* **2020**, *15*, 19–28.
20. Winkelhaus, S.; Grosse, E.H. Logistics 4.0: A systematic review towards a new logistics system. *Int. J. Prod. Res.* **2020**, *58*, 18–43. [CrossRef]
21. Hofmann, E.; Rüsch, M. Industry 4.0 and the current status as well as future prospects on logistics. *Comput. Ind.* **2017**, *89*, 23–34. [CrossRef]
22. Business2Community. Issues in Warehouse Management Systems. 2018. Available online: https://www.business2community.com/product-management/top-5-warehouse-management-problems-solve-02027463 (accessed on 17 July 2018).
23. Poon, T.; Choy, K.; Chow, H.K.; Lau, H.C.; Chan, F.T.; Ho, K. A RFID Case-based Logistics Resource Management System for Managing Order-Picking Operations in Warehouses. *Expert Syst. Appl.* **2009**, *36*, 8277–8301. [CrossRef]
24. Richards, G. *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse*; Kogan Page Publishers: London, UK, 2017.
25. Ud Din, F.; Henskens, F.; Paul, D.; Wallis, M. Agent-Oriented Smart Factory (AOSF): An MAS Based Framework for SMEs Under Industry 4.0. *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 44–54.
26. Ud Din, Fareed. Agent Oriented Smart Factory (AOSF): A MAS Based Framework for SMEs under Industry 4.0. Ph.D. Thesis, The University of Newcastle, Callaghan, NSW, Australia, 2020; pp. 67–86.

27. Ud Din, F.; Henskens, F.; Paul, D.; Wallis, M. Formalisation of Problem and Domain Definition for Agent Oriented Smart Factory (AOSF). In Proceedings of the 2018 IEEE Region Ten Symposium (Tensymp), IEEE, Sydney, NSW, Australia, 4–6 July 2018; pp. 265–270.
28. Ud Din, F.; Paul, D.; Ryan, J.; Henskens, F.; Wallis, M. Revitalising and Validating the Novel Approach of xAOSF Framework Under Industry 4.0 in Comparison with Linear SC. In *Agents and Multi-Agent Systems: Technologies and Applications 2020*; Springer: Singapore 2020; pp. 3–16.
29. Ud Din, F.; Paul, D.; Ryan, J.; Henskens, F.; Wallis, M. Validating Time Efficiency of AOSR 2.0: A Novel WMS Planner Algorithm for SMEs, under Industry 4.0. Available online: http://www.jsoftware.us/vol15/415-MC3026.pdf (accessed on 4 June 2021).
30. Ud Din, F.; Paul, D.; Ryan, J.; Henskens, F.; Wallis, M. Conceptualised Visualisation of Extended Agent Oriented Smart Factory (xAOSF) Framework with Associated AOSR-WMS System. Available online: http://www.jsoftware.us/index.php?m=content&c=index&a=show&catid=228&id=3040 (accessed on 4 June 2021).
31. Preuveneers, D.; Berbers, Y. Modeling Human Actors in an Intelligent Automated Warehouse. In *International Conference on Digital Human Modeling*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 285–294.
32. Java Agent Development Framework. JADE Open Source Project: Java Agent Development Environment Framework. 2017. Available online: http://jade.tilab.com/ (accessed on 18 August 2017).
33. Rodriguez Masdefiol, M.d.M.; Stävmo, F. Industry 4.0–Only designed to fit the German automotive industry: A multiple case study on the feasibility of Industry 4.0 to Swedish SMEs. 2016. Available online: https://www.diva-portal.org/smash/get/diva2:933316/FULLTEXT01.pdf (accessed on 18 November 2019).
34. Pech, M.; Vrchota, J.; Bednář, J. Predictive Maintenance and Intelligent Sensors in Smart Factory. *Sensors* **2021**, *21*, 1470. [CrossRef]
35. Popescu, C.K.; Oasa, R.-S.; Geambazi, P.; Alexandru, B. Real-Time Process Monitoring, Industry 4.0 Wireless Networks, and Cognitive Automation in Cyber-Physical System-based Manufacturing. *J. Self Gov. Manag. Econ.* **2021**, *9*, 1.
36. Jones, M.M.; Juneja, M.O.; Gnanamurthy, K.; Kandikuppa, K.; Sheu, J.Y.W.; William, E.R.V.; Hadagali, G.R.; Rawat, S.S.; Berry, V.; Agrawal, D.; et al. Consigned Inventory Management System. U.S. Patent 14,499,372, 2016.
37. EuroSped. Dataset Information for Warehouse and Logistics. 2018. Available online: http://www.eurosped.bg/en/eurolog-warehouse-logistics-4pl/ (accessed on 25 July 2018).
38. Global, D.G.I. Warehouse Prducts Classes. 2018. Available online: http://www.dgiglobal.com/classes (accessed on 25 July 2018).
39. Russell, S.; Norvig, P.; Intelligence, A. A modern approach. *Artif. Intell. Prentice Hall Englewood Cliffs* **1995**, *25*, 27.
40. FIPA ACL. FIPA ACL Message Structure Specification. Foundation for Intelligent Physical Agents. 2002. Available online: http://www.fipa.org/specs/fipa00061/SC00061G (accessed on 30 June 2004).
41. Chen, J.C.; Cheng, C.H.; Huang, P.B.; Wang, K.J.; Huang, C.J.; Ting, T.C. Warehouse Management with Lean and RFID Application: A Case Study. *Int. J. Adv. Manuf. Technol.* **2013**, *69*, 531–542. [CrossRef]
42. Li, L. *Supply Chain Management: Concepts, Techniques and Practices: Enhancing Value Through Collaboration*; World Scientific Publishing Co Inc.: Singapore, 2007.
43. Demo3D-Simulation-Tool. Demo3D, A Product of SOLIDWORKS and Rockwell Automation Industry, USA, for Simulating Industrial Setups. 2019. Available online: https://www.demo3d.com/home/ (accessed on 31 May 2019).
44. Rockwell-Automation. Rockwell Automation Industry, USA. 2019. Available online: https://www.rockwellautomation.com/site-selection.html (accessed on 31 May 2019).
45. Warehousing, G.O. GrayOrange Autonomous Warehousing. 2018. Available online: https://www.greyorange.com/butler-goods-to-person-system (accessed on 25 July 2018).
46. Warehousing, U. Information of Warehouse Product Management. 2018. Available online: https://www.unleashedsoftware.com/product/inventory/product-management (accessed on 25 July 2018).
47. Grant, E. Big Data-driven Innovation, Deep Learning-assisted Smart Process Planning, and Product Decision-Making Information Systems in Sustainable Industry 4.0. *Econ. Manag. Financ. Mark.* **2021**, *16*, 9–19.
48. Coatney, K.; Poliak, M. Cognitive Decision-Making Algorithms, Internet of Things Smart Devices, and Sustainable Organizational Performance in Industry 4.0-based Manufacturing Systems. *J. Self Gov. Manag. Econ.* **2020**, *8*, 9–18.