Transport Reduction in a Production Grid

Leo van Moergestel, Erik Puik and Daniël Telgen
Department of Computer science
HU Utrecht University of Applied Sciences
Utrecht, the Netherlands
Email: leo.vanmoergestel@hu.nl
John-Jules Meyer
Intelligent systems group Utrecht University
Utrecht, the Netherlands
Email: J.J.C.Meyer@uu.nl

***Abstract*—In a production environment where different products are being made in parallel, the path planning for every product can be different. The model proposed in this paper is based on a production environment where the production machines are placed in a grid. A software entity, called product agent, is responsible for the manufacturing of a single product. The product agent will plan a path along the production machines needed for that specific product. In this paper, an optimization is proposed that will reduce the amount of transport between the production machines. The effect of two factors that influence the possibilities for reductions is shown in a simulation, using the proposed optimization scheme. These two factors are the redundancy of production steps in the grid and the number of steps where the order of execution is irrelevant. This paper presents for certain classes of production situations a method to reduce the number of transport hops between the production machines.**

Keywords-Multiagent-based manufacturing; production path planning.

I. INTRODUCTION

The production industry has had several revolutions. The first revolution was the use of steam power to facilitate production. The second revolution was the introduction of production lines based on the use of electrical energy and resulting in mass production. The rise of computer technology resulted in the third revolution. Many production tasks were automated, programmable logic controllers (PLC), distributed control systems (DCS) and robots were introduced on the production floor. The latest revolution is the integration of information technology in the production process as a whole. This has been described by the term industry 4.0 or cyber physical systems. The effect is that the requirements for manufacturing are rapidly changing due to newly arrived technologies like 3D-printing and end-user involvement using Internet technology.

At the Utrecht University of Applied Sciences, research is done on agile manufacturing. The aim is to achieve low-cost production of small quantities or even single user-specified products. This means that hardware, as well as software should be developed to make this possible. The work in this paper is based on a paper presented at the Intelli 2015 conference [1] and other previous work. The hardware that has been developed are cheap reconfigurable devices, called equiplets. Equiplets consist of a basic platform on which specific front-ends can be attached. When a front-end is attached to an equiplet, it will be capable to perform one or several specific production steps [2].

The software that is used in the production environment, is based on multiagent technology [3]. An agent is an autonomous software entity, having responsibilities and playing a role in the whole manufacturing software infrastructure. Two specific agent roles are the basis of the manufacturing system. The role and responsibility to have a single product made is assigned to a product agent. The role to control an equiplet and to offer production steps is assigned to an equiplet agent [4].

This paper will focus on the product agents and specifically the planning part of its role in the manufacturing. The main goal in this production process is to minimize the flow cost of products among the machines. Section II will show an overview of the roles and responsibilities of the product agent. This will also reveal the manufacturing concept used in our research. In Section III the definition of terms used in the paper are introduced and explained. After the introduction of the terms, the path planning approach is the topic of Section IV. To test this approach, the implementation has been tested in a simulated environment. The results of these simulations are given in Section V on results and discussion. In Section VI, related work is discussed among other work that is related to this new manufacturing paradigm. A conclusion and bibliography will end the paper.

II. AGENT-BASED MANUFACTURING

In the previous section, the concepts of product agent and equiplet agent have been introduced. The manufacturing concept will now be discussed. Industry 4.0 is also characterised as a cyber physical system. In this section these two parts will be explained starting with the physical aspect.

Figure 1. An equiplet

A. Physical aspect

As stated in the introduction, the actual production is done by so-called equiplets. The equiplets are placed in a grid topology for reasons that will be explained at the end of this section. Each equiplet offers one or more production steps and by combining a certain set of production steps, a product can be made. The set of production steps that can be performed, depends on the type of front-end that is attached to the equiplet. This way every equiplet acts as a reconfigurable manufacturing system (RMS) [5]. In Figure 1 an equiplet is shown. This equiplet is a pick-and-place machine based on a delta-robot mechanism. Agent technology opens the possibilities to let a grid of these equiplets operate and manufacture different kinds of products in parallel, provided that the required production steps are available [4].

B. Cyber aspect

Equiplets are represented by an equiplet agent. Every product requires a given set of production steps and the equiplets in the grid should implement these steps to make it possible to manufacture a specific product. Every product has its own software entity or product agent that is responsible for the manufacturing of a single product. By letting this product agent interact with the equiplet agents the actual manufacturing will take place. In the grid, more than one product agent can be active at any moment, so different products can be made in parallel.

For a product to be made, a sequence of production steps has to be done. More complex products need a tree of sequences, where every sequence ends in a half-product or part, needed for the end product. As a software representative of the equiplet, the equiplet agent advertises its capabilities as production steps on a blackboard that is available in a multiagent system where also the product agents live. A product agent is responsible for the manufacturing of a single product and knows what to do, the equiplet agents know how to do it. A product agent selects a set of equiplets based on the production steps it needs and tries to match these steps with the steps advertised by the equiplets. This selection of equiplets is called the planning and scheduling phase. The planning and scheduling of a product is an atomic action, done by the product agent in cooperation with the equiplet agents and takes several steps [6]. The planning and scheduling is atomic to prevent problems that arise if more product agents want to schedule steps on equiplets at the same time. If one agent is planning and scheduling, other newly arriving product agents have to wait until the agents finishes the allocation of equiplets. Let us assume that a single sequence of steps is needed.

1) From the list of production steps, the product agent builds a set of equiplets offering these steps;
2) The product agent will ask the equiplets involved about the feasibility and duration of the steps; Actually the equiplet agent will run a simulation of the step required using the parameters given, to check the feasibility and duration of the step.
3) Next the product agent will generate a path along equiplets;
4) The product agent will schedule the product path using first-fit (take the first opportunity in time for a production step) and a scheduling scheme known as earliest deadline first (EDF) [6];
5) If the schedule fails, the product agent reports this to the user and proposes a later production time if possible.

For more complex products, consisting of a tree of sequences, the product agent spawns child agents that are each responsible for a single sequence. A child agent has the same functionality as the parent agent, but is only responsible for a subset op production steps. The parent agent is in control of its children and acts as a supervisor. It is also responsible for the last single sequence of the product. In Figure 2, the first two half products are made using step sequences $< \sigma_1, \sigma_2 >$ and $< \sigma_3, \sigma_4 >$. These sequences are taken care of by child agents, while the parent agent will complete the product by performing the step sequence $< \sigma_4, \sigma_7, \sigma_2, \sigma_1 >$. It means that every single product agent, child or parent, has only a single sequence of steps to perform by itself.

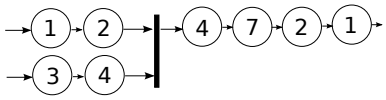

Figure 2. Manufacturing of a product consisting of two half-products

Some important features of the manufacturing model are:

- Every product agent is responsible for only one product to be made;
- The requests for products arrive at random;

- Every product will have its own path along the equiplets during manufacturing;
- The product agent will guide the product along the equiplets.

In the final implementation, a webinterface helps the end-user to design his/her specific product [2]. At the moment all features are selected, a product agent will be created. Because every product can have a different walk along the equiplets, the equiplets are in a grid arrangement that turns out to be more efficient than a line arrangement as used in batch processing [7].

III. STEP PATH AND PRODUCT PATH

This section will define the concepts step path and product path. In Subsection III-A, step path classes will be introduced and in Subsection III-B special cases of step paths are discussed.

Consider a situation where a product is built by 11 production steps. Let us assume that we have 3 equiplets A, B and C. Equiplet agent A offers production step set $E_A = \{1, 2, 3, 4, 8\}$, $E_B = \{5, 6, 7\}$ and $E_C = \{9, 10, 11\}$. The product agent representing our 11-steps product will choose equiplet A first to perform steps 1, 2, 3 and 4. Next equiplet B is used to perform steps 5, 6 and 7. Then, we need again equiplet A for step 8 and finally equiplet C for the last tree steps 9, 10 and 11. This so called *step path* is visualized in Figure 3.

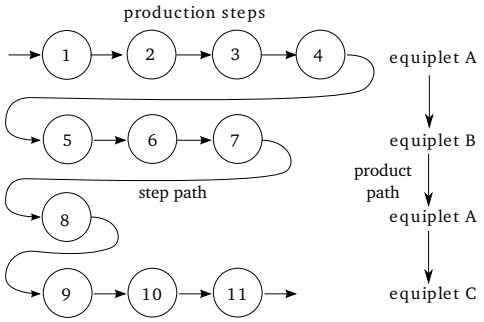

Figure 3. Step path and product path

Definition 1 (Step path)**.** *A step path is a path along a sequence of production steps that a product agent has to follow to complete a product.*

In the example that is visualised in Figure 3 where the step path is shown, another path emerges. This is the path along the equiplest involved. In case of the example, it is a path from equiplet A to equiplet B, from equiplet B to A and finally from equiplet A to equiplet C. This type of path will be referred to as *product path.*

Definition 2 (Product path)**.** *A product path is a path along a sequence of equiplets that a product agent has to follow to complete a product.*

A. Step path classes

In the previous example of our 11-step product (Figure 3) the production steps are in line so our path is a single thread. Figure 4 shows the two possibilities that are considered in this paper: a single line and a tree structure where two half-fabricates are combined.

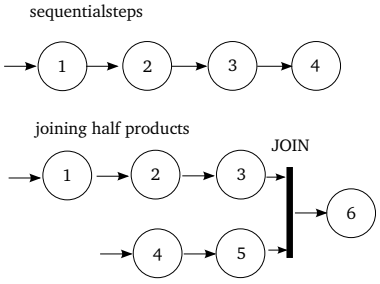

Figure 4. Two combinations of sequential production steps

When these product paths as shown in Figure 4 are written in sets (using: $\{...\}$) and tuples (using: $< ... >$) this results in:

- Single path, with tuple notation for a fixed order of steps: $< \sigma_1, \sigma_2, \sigma_3, \sigma_4 >$
- Joining half products: $< \{< \sigma_1, \sigma_2, \sigma_3 >, < \sigma_4, \sigma_5 >\}, \sigma_6 >$

B. Special cases of step paths

In some situations, the order of steps is irrelevant. This results in several possibilities for the step paths. Only one path of these possibilities should be chosen and the number of possibilities is $n!$ in case we have n steps with irrelevant order. This situation can be seen in Figure 5.

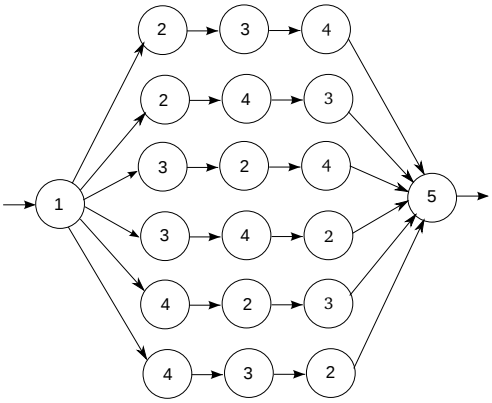

Figure 5. A set of steps with irrelevant order

In formula, this means that the set-notation is used for the steps with irrelevant order:

$$< \sigma_1, \{\sigma_2, \sigma_3, \sigma_4\}, \sigma_5 > \quad (1)$$

Parallelism can be achieved if the product has a tree structure as in Figure 6. On the left side of this figure, four incoming

arrows, each denoting the start of a production path can be seen. Each path will construct a subpart for the final product and because these paths are independent, these subparts can be made in parallel. At every join in the figure these sub-parts are combined to be input to the next step or steps.

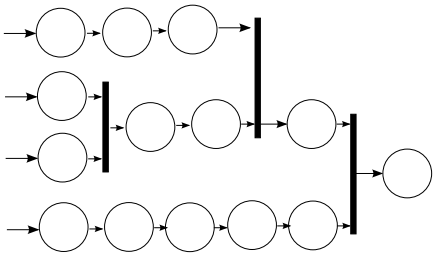

Figure 6. A tree of steps

In the situation of a tree structure a collection of product agents for a single product will be used. The start situation will be one agent, but this agent will spawn child-agents for the separate tuples. The parent agent is in control of its children and acts as a supervisor. It is also responsible for the last single sequence of the product. In Figure 6, we start at the right-hand side and walk backwards to the beginning of the production on the left. At every join child-agents will be created. The parent will wait for its children to complete their subpart. This will be done for every join and will be repeated until the start of the tree structure. When all agents succeed in planning and scheduling, the production will start. At every join the child agents are absorbed by the waiting agent, taking over the collected information and continuing the path until the end is reached as a single agent.

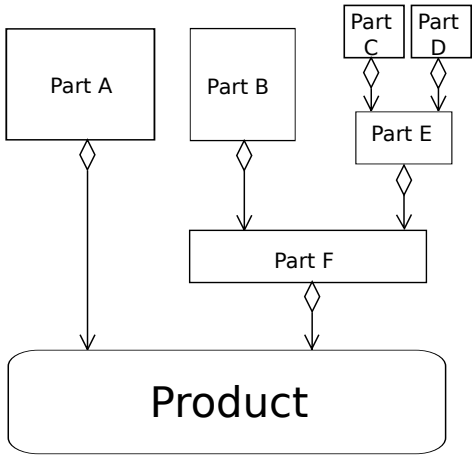

Figure 7. Product consisting of subparts

This situation arises many times, because most products consist of subparts (Figure 7). The product agent at the root of the tree will finally collect all information from its children. The effect of this decomposition of complex products is that every product agent only has to deal with a single tuple of production steps. The relationship between these product agents is the fact that they are working on the same product.

IV. Path planning

A product agent should plan a path along the equiplets. This path will depend on the product steps to be done and the equiplets involved. In this section, a graph-based model of the production is presented followed by matrix-based representations. These matrix-based representations will be used in the optimization system that is the main subject of this paper.

A. Graph representation

The production system can be represented using special classes of graphs, such as a bipartite graph and a tripartite graph.

Definition 3 (Bipartite graph)**.** *A bipartite graph is defined as a triple $G = (V_1, V_2; E)$ where V_1 and V_2 are two disjoint finite sets of vertices and $E = \{(i_k, j_k) : i_k \in V_1, j_k \in V_2; k = 1...d\}$ is a set of edges. Let $|V_1| = m$ and $|V_2| = m$.*

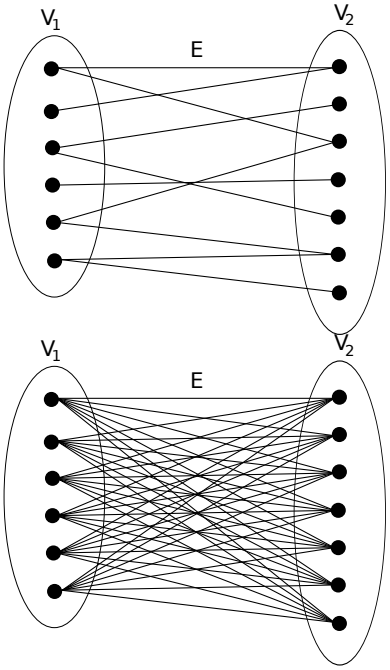

Figure 8. Bipartitegraph and complete bipatitegraph

If all vertices of V_1 have edges to all vertices of V_2 the graph is called a complete bipartite graph. If $|V_1| = m$ and $|V_2| = n$ this is denoted by $K(m, n)$.

Definition 4 (Product set)**.** *Let V_1 and V_2 be two sets. The product set of V_1 and V_2 is the set of all ordered pairs (i, j) such that $i \in V_1$ and $j \in V_2$. This is written as $V_1 \times V_2$*

By definition of the product set, it means that a bipartite graph is complete if $E = V_1 \times V_2$. The product set is also called Cartesian product.

Definition 5 (Tripartite graph)**.** *A tripartite graph is defined as a quintuple* $G = (V_1, V_2, V_3; E_1, E_2)$ *where* V_1 *and* V_2 *and* V_3 *are three disjoint sets of vertices and* $E_1 = \{(i_k, j_k) : i_k \in V_1, j_k \in V_2; k = 1...d_1\}$ *and* $E_2 = \{(j_n, h_n) : j_n \in V_2, h_n \in V_3; n = 1...d_2\}$ *are two sets of edges. Let* $|V_1| = m$ *and* $|V_2| = m$*.*

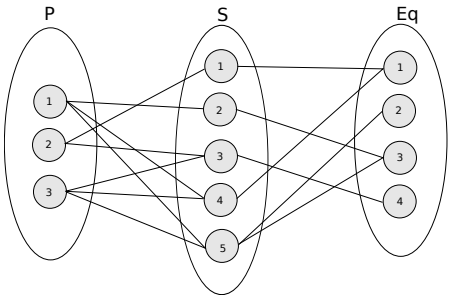

Figure 9. A tripartite graph as it occurs in the production system

In Figure 9 the situation is shown for the agile production system. On the left side the products to be made are in set P. In the middle, the set S of steps is displayed and on the right the set Eq of equiplets. Edges show the connection between the products and the steps as well as the steps and the equiplets. The step path for product P_1 is:

$$< \sigma_5, \sigma_2, \sigma_4 >$$

There are four equiplets, where Equiplet 1 offers steps σ_1 and σ_4, Equiplet 2 offers step σ_5, Equiplet 3 offers steps σ_2 and σ_5 and Equiplet 4 offers step σ_3.

B. Step matrix

Consider a grid G of N equiplets, together offering M production steps, this grid can be described by a matrix. This matrix is called *step matrix*.The step matrix G_{step} shows the mapping of equiplets to production steps.

$$G_{step} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{pmatrix} \quad (2)$$

In this matrix, $a_{ij} = 1$ if equiplet E_i offers step σ_j, otherwise $a_{ij} = 0$.

C. Optimization

A step path is a sequence of production steps. For instance, consider a product to be built with three production steps, this product has step path:

$$< \sigma_5, \sigma_2, \sigma_4 > \quad (3)$$

Let us assume a simple grid with four equiplets E_1, E_2, E_3 and E_4, each offering a set of steps. The steps offered by an equiplet are denoted between parentheses as in $E_1(\sigma_1, \sigma_4)$. This grid can be described by this set of equiplets:

$$\{E_1(\sigma_1, \sigma_4), E_2(\sigma_5), E_3(\sigma_2, \sigma_5), E_4(\sigma_3)\} \quad (4)$$

This situation can also be described by the step matrix G_{step}.

$$\begin{array}{c|cccc} & E_1 & E_2 & E_3 & E_4 \\ \sigma_1 & 1 & 0 & 0 & 0 \\ \sigma_2 & 0 & 0 & 1 & 0 \\ \sigma_3 & 0 & 0 & 0 & 1 \\ \sigma_4 & 1 & 0 & 0 & 0 \\ \sigma_5 & 0 & 1 & 1 & 0 \end{array} \quad (5)$$

A product agent will make a selection of these equiplets based on the production step or steps that must be performed to construct the product. Next, the product agent will ask the equiplet if the steps offered are feasible given the parameters for the steps. The positive response from the equiplet agent contains an estimated time to complete a given step. This information about the duration of a step will be used in the scheduling phase. When a negative response is received by the product agent it will discard the equiplet. Several solutions to map the steps to equiplets may exist. A sufficient solution for the given situation with a minimum of transitions is:

$$< E_3(\sigma_5), E_3(\sigma_2), E_1(\sigma_4) > \quad (6)$$

To find an efficient solution, we try to minimise the transitions or hops between different equiplets. This is done by using a so called production matrix G_p. This production matrix can be derived from the step matrix by selecting the rows of the production steps in the same order as in the tuple that describes the step path $< \sigma_5, \sigma_2, \sigma_4 >$.

$$\begin{array}{c|cccc} & E_1 & E_2 & E_3 & E_4 \\ \sigma_5 & 0 & 1 & 1 & 0 \\ \sigma_2 & 0 & 0 & 1 & 0 \\ \sigma_4 & 1 & 0 & 0 & 0 \end{array} \quad (7)$$

The production matrix can be reduced by eliminating the columns that contain only zeros. This means that the equiplet on top of this column is not involved in the production of this specific product. In this case the column under E_4 will be removed. This results in a matrix (8) where for every σ_i in this step path a row of a production matrix is created:

$$\begin{array}{c|ccc} & E_1 & E_2 & E_3 \\ \sigma_5 & 0 & 1 & 1 \\ \sigma_2 & 0 & 0 & 1 \\ \sigma_4 & 1 & 0 & 0 \end{array} \quad (8)$$

The rows have the same order as the sequence of steps. Matrix element α_{ij} gives the relation between equiplet E_j and production step σ_x at row i. If the step σ_x at row i is supported by equiplet E_j then $\alpha_{ij} = 1$. Not supported steps result in $\alpha_{ij} = 0$.

Optimization should result in a new matrix, that will be called the path matrix, where α_{ij} has a slightly different meaning and can be different from 1 or 0, giving the product agent a clue for its choice. The product agent will choose the equiplet corresponding with the highest value of α_{ij}. The

optimization is minimizing the transitions for a product from equiplet to equiplet. The optimizing algorithm will search for columns j with sequences of $\alpha_{ij} = 1$ and increment the values in a given sequence by the length of the sequence minus one. This will be done for all columns starting with $\alpha_{ij} = 1$. The matrix of the example has a column under E_3 with a length of 2, with the result that the values of this sequence will be incremented by 1. The production matrix transforms to the path matrix (9):

$$\begin{array}{cccc} & E_1 & E_2 & E_3 \\ \sigma_5 & 0 & 1 & 2 \\ \sigma_2 & 0 & 0 & 2 \\ \sigma_4 & 1 & 0 & 0 \end{array} \quad (9)$$

A value higher than 1 means that more steps can be done in sequence on the same equiplet. Based on this matrix, the product agent will choose equiplet E_3 for steps σ_5 and σ_2. The path matrix can be cleaned up by changing values that will not be used in the path to zero.

$$\begin{array}{cccc} & E_1 & E_2 & E_3 \\ \sigma_5 & 0 & 0 & 2 \\ \sigma_2 & 0 & 0 & 2 \\ \sigma_4 & 1 & 0 & 0 \end{array} \quad (10)$$

The optimization algorithm works stepwise and can be completely automatised. First the best starting point is searched for. This will reveal the best equiplet(s) to start with. Let us assume that we have n steps in the step path. This results in a production matrix of n rows. Suppose that the algorithm reveals a maximum set of k steps to be completed by one equiplet as a start. This means that after completing this sequence of k steps, $n - k$ rows, representing $n - k$ steps, should still be done. We reached this point of $n - k$ steps to be done, with the minimum of movements of the product between equiplets. The algorithm is applied to the remaining part (the $n - k$ rows) of the production matrix, without taking into account the previous k rows. We reach a new situation where the number of rows is again reduced. This is repeated until the number of remaining rows is 0. Because of the fact that after every iteration we reach a situation with the minimum of movements of the product between equiplets, the final situation, where the number of rows to be done is 0, will also be reached with the minimum of movements.

D. Region with irrelevant order of steps

Now, consider the situation where there exists a region in the production matrix where the order of steps is irrelevant. This region will be referred to as a region with irrelevant step order. If this irrelevant step order region concerns the whole production matrix, there are no borders with a region where the order is fixed as discussed before. We will discuss this situation first and next a situation where the irrelevant step order region is embedded in two regions with fixed order.

When there are no borders with other regions, the used approach is the following: generate a vector v from the matrix where we sum all separate columns. This means for element v_j of vector v, assuming a matrix with N rows:

$$v_j = \sum_{i=1}^{i=N} \alpha_{ij} \quad (11)$$

From this vector the highest value will be chosen as a start. The irrelevant region will decrease by v_j and a new vector will be generated for the remaining smaller region until all steps needed are taken into account.

When there are borders, a slightly different approach will be used. At the border at the top of the irrelevant step order region, there should be a sequence of at least one step resulting from the fixed step order region. In this case a search will be done to find the best match with this already available sequence from the previous region. The same approach holds for the region at the bottom. A special case in this situation could be a sequence that has the size of the special region. Such a sequence will be called a tunnel and special care should be taken. If there are no matches at the upper or lower border, first matching sequences should be investigated. Matching sequences will not introduce a hop and if these matching sequences at top of border do not cover the whole special region, the tunnel can eventually be used introducing two hops, but if the matching sequences on top and bottom together cover the region only one hop is needed.

Two caveats should be mentioned here. If the boundary with a fixed region has more than one maximum (that should be equal of course), these possibilities should be investigated for the best fit. This means we have to look for the maximum in the fixed region that can be extended to the longest sequence by adding a member of the vector v. The number of maxima (not the maxima itself) will give a clue about where to start, at the top or the bottom border. This will be shown in an example.

Another caveat has to do with overlapping sequences in irrelevant step order region. Consider the situation for a irrelevant region depicted in the matrix (12):

$$\begin{array}{cccc} & E_1 & E_2 & E_3 \\ \sigma_a & 1 & 0 & 0 \\ \sigma_b & 1 & 1 & 0 \\ \sigma_c & 0 & 1 & 1 \\ \sigma_d & 0 & 0 & 1 \end{array} \quad (12)$$

Generating the vector v will result in $(2, 2, 2)$. However, the choice to be made depends on the next vector that would result from this choice. If the middle maximum is chosen, the resulting vector v is $(1, 0, 1)$ resulting in a total of two transitions or hops. If the selection was for the first maximum the resulting v would be $(0, 1, 2)$, while choosing the last maximum v would be $(2, 1, 0)$. Both of the latter situations result in only one hop.

As an extra example of the approach discussed so far, consider the situation shown in Figure 10. At the top are two maxima having a value of 3, resulting from the evaluation of the previous fixed order region. The bottom has only one maximum, also resulting from the evaluation of the following fixed order region. The vector for the irrelevant order region is:

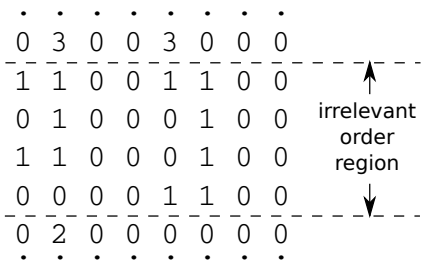

Figure 10. Border situations

$(2, 3, 0, 0, 2, 4, 0, 0)$. Because the bottom border has the least number of maxima, we start there to fit with the vector values. If we started at the top, we could choose the first fit, but then we would loose the fitting possibility at the bottom. If the fitting at the bottom is performed, only one row is left to be handled, having vector $(0, 0, 0, 0, 1, 1, 0, 0)$. This vector fits with one of the maxima at the top border resulting in matrix (13), having only one hop.

$$\begin{matrix} . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \end{matrix} \quad (13)$$

E. Alternative paths

Having a path with a minimum of hops is nice, but in some cases it might by handy to have some alternatives at hand. To generate alternatives with the same number of hops, two possibilities can be used. When these possibilities are combined, a total number of four optimum paths can be calculated. However, it is not guaranteed that these paths are different. By using simple examples these approaches will be demonstrated. Consider the matrix (14), having no special region.

$$\begin{matrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix} \quad (14)$$

If we apply the optimization algorithm starting from top to bottom and choosing the maximum most on the left, it will result in matrix (15):

$$\begin{matrix} 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \quad (15)$$

An alternative can be found by starting at the bottom, but again choosing the maximum most to the left. Now we get matrix (16):

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \quad (16)$$

Possibility number three can be found by again starting at the top, but now choosing the maximum most to the right. The result is shown in matrixx (17):

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix} \quad (17)$$

The fourth possibility can be found by starting at the bottom and selecting the maximum most on the right, resulting in matrix (18):

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{matrix} \quad (18)$$

An interesting aspect is looking at the percentage or number of equiplets that overlap. By this is meant the number of equiplets that are used in two alternatives. Table I shows this overlap for the four different solutions found. This overlap is very dependant on the initial matrix given, but from this example it becomes clear that the smallest overlap and perhaps the one of the best alternatives to our first solution is solution four where both strategies (starting top/bottom, choosing right/left) are changed.

Two remarks should be made at this point. First, when there are more than two maxima to choose from, only the maxima most on the left or most on the right will be chosen in the

TABLE I. percentage of overlap

	1	2	3	4
1	100	25	50	0
2	25	100	0	48
3	50	0	100	25
4	0	48	25	100

approach presented here. The idea of generating alternatives is to find another path with an equal numbers of hops, not to find all alternative paths with an equal number of hops. The algorithm should be fast and simple, as it is now. An alternative path should be considered if the scheduling constraints, like deadline or load of a certain equiplet give rise to looking for an alternative solution.

The second remark is that it is clear that alternatives in this situation (with a fixed sequence of steps), are only possible if steps are available at multiple equiplets. However, in case of a production grid, it is a good design strategy if equiplets are duplicated or that a certain step is available at more than one equiplet. This will prevent the existence of a single point of failure and will make the grid production platform more reliable.

V. TEST RESULTS DISCUSSION

In this section, a description of the software implementation, the results, the time it takes to run the optimisation and discussion are presented.

A. Software implementation

To implement the optimisation algorithm, several command line tools running under Linux were developed. These tools were written in C, resulting in fast and compact applications. The reason for choosing Linux was based on the fact that the equiplet software and the agent platform for production, is also Linux-based.

a) Generate production matrix: The first tool developed was a tool to generate a production matrix. The parameters used are the dimension of the matrix and the number of one's per row. An output example with 5 rows, 8 columns and 4 one's per column looks like:

```
0 0 0 1 1 0 1 1
0 0 1 0 1 1 0 0
0 0 1 0 1 0 0 1
0 0 1 0 0 0 1 0
0 0 1 0 0 0 0 1
```

Because of the use of random numbers for determining the position of one's, many different matrices can be generated, given the same parameters for the dimensions.

b) Analyse the matrix: The first step to get to an optimum path is generated by the tool *analyse*. Given the previous matrix, *analyse* will count the sequences of one's in a column and adjust the numbers accordingly, resulting in:

```
0 0 0 1 3 0 1 1
0 0 4 0 3 1 0 0
0 0 4 0 3 0 0 1
0 0 4 0 0 0 1 0
0 0 4 0 0 0 0 1
```

c) cleanup: The result of *analyse* should be adjusted to get an optimum path. The tool *cleaunup* is used to adapt the matrices generated by analyse It will start from the beginning of a matrix and remove all irrelevant options, turning them to zero's. Another thing that will be done is adjusting the value for overlapping paths. If the highest value for a column is not unique, it will choose the one most to the left, this is the first one found.

```
0 0 0 0 3 0 0 0
0 0 0 0 3 0 0 0
0 0 0 0 3 0 0 0
0 0 0 2 0 0 0 0
0 0 0 2 0 0 0 0
```

d) Handle special region: The tool *special* is used to adapt the matrices generated by *analyse* and *cleanup*. The first thing that will be done is counting the total of one's in the special region per column. Next it will look for the best fit with the upper part, as well as a fit with the lower part. An adjustment will be made to the fitting parts found so far. Finally *special* will fill up the missing area with the best choice available and continue until the special region is completed

e) Tools for generating alternatives: To generate alternatives and still using the previous tools that work from top to bottim and left to right, two other simple tools have been developed. The tool *mirror* will generate a matrix that has a mirrored sequence of columns. For our example production matrix this looks like:

```
1 1 0 1 1 0 0 0
0 0 1 1 0 1 0 0
1 0 0 1 0 1 0 0
0 1 0 0 0 1 0 0
1 0 0 0 0 1 0 0
```

Using *mirror* twice will result in the original matrix again. The tool *upsidedown* will generate a matrix where the sequence of rows is inverted. For our example matrix this will look like:

```
0 0 1 0 0 0 0 1
0 0 1 0 0 0 1 0
0 0 1 0 1 0 0 1
0 0 1 0 1 1 0 0
0 0 0 1 1 0 1 1
```

Of course *mirror* and *upsidedown* can be combined resulting in matrix:

```
1 0 0 0 0 1 0 0
0 1 0 0 0 1 0 0
1 0 0 1 0 1 0 0
0 0 1 1 0 1 0 0
1 1 0 1 1 0 0 0
```

Using these two tools, the previous mentioned tools can still be used without any adaptation. The matrices are changed and the result can be put in the right order again.

B. Results

To test the optimising approaches discussed in the previous section, test sets have been generated. All these sets consist of 8 matrices. Thus, 8 equiplets are assumed and the production requires 32 steps. First the effect of redundancy is investigated. This is done by using test sets where at every row, there are 1, 2, 3 or 4 choices for equiplets to perform a certain step. The results of using the optimization approach are shown in Figure 11. In this figure, a slight decrease in the number of hops is shown. This is expected due to the fact that higher redundancy gives rise to longer sequences of steps on the same equiplet, thus reducing the number of hops.

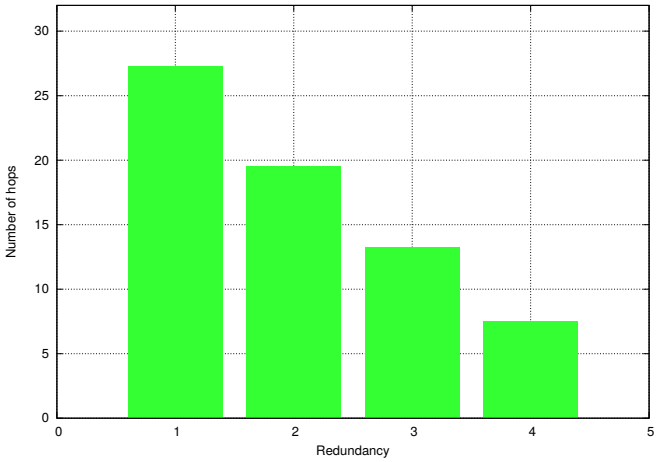

Figure 11. The effect of redundancy

Next, the effect of the size of the region where the order is irrelevant is investigated. This is done by using the same test sets that have been used to see what the effect of redundancy is. The size of the region where the order of steps is irrelevant, changes from 0 (no special region) until 32 (the whole matrix is a special region) in steps of 4. The special region is always placed in the middle of the matrix. The results for the test sets where the redundancy is only 1, are shown in Figure 12. In the subsequent Figures 13, 14 and 15 the results are shown for test sets having a redundancy of 2, 3 and 4. In all figures a decrease of the number of hops can be seen. This is also a result that is expected, because a region where the order of steps is irrelevant opens more possibilities to generate longer sequences of steps on the same equiplet.

C. Speed

This section will give an impression of the speed of the optimisation. The simulation where the optimisation was used, was run on a standard desktop system with an Intel(R) Core(TM)2 Duo CPU with a 2.33GHz clock and a 4096 KB cache. The system had 4GB of memory. The operating system was Linux (Ubunto 14.04 LTS). The command *time* was used to measure the execution time of the simulation containing the optimisation. The results for a manufacturing environment with 8 equiplets and 32 production steps were:

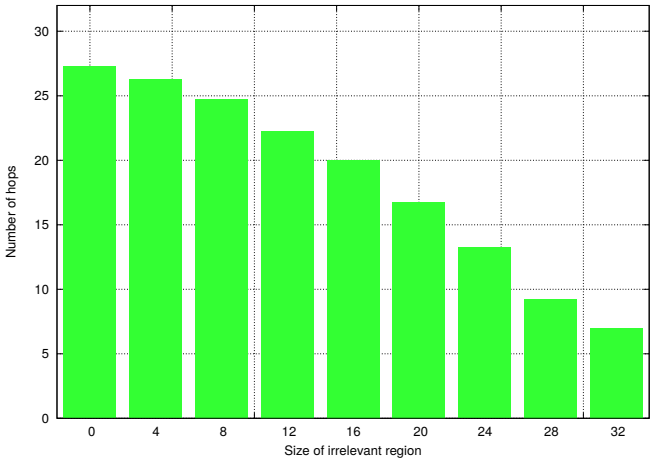

Figure 12. The effect of the size of the irrelevant step region

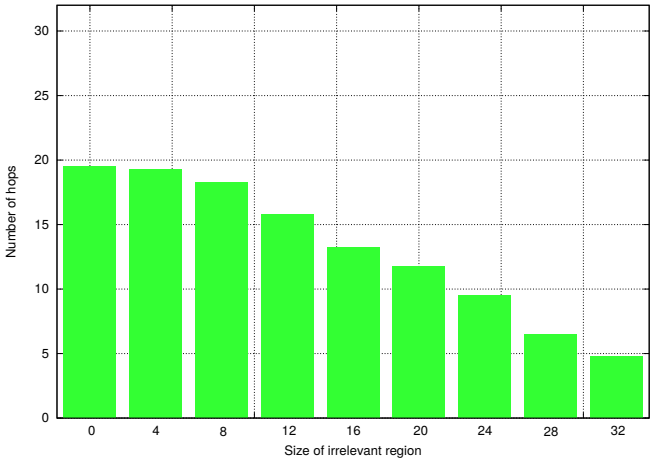

Figure 13. The effect of the size of the irrelevant step region

```
real 0m0.225s
user 0m0.003s
sys 0m0.007s
```

This shows that the actual computing time is actually very short in comparison to the transport time, where the time for a hop might be in the order of seconds.

D. Discussion

The test sets were used to test the optimising approach presented in the previous section and this approach turns out to work in the given situations. Important is to emphasize that reduction in transition or hops is an important optimization for the grid production paradigm. In [7], a transport system for the grid is described. This transport system is based on the use of automated guided vehicles (AVG). It turned out that the transport system becomes the bottleneck in a manufacturing grid if the production steps are relatively short, which is in

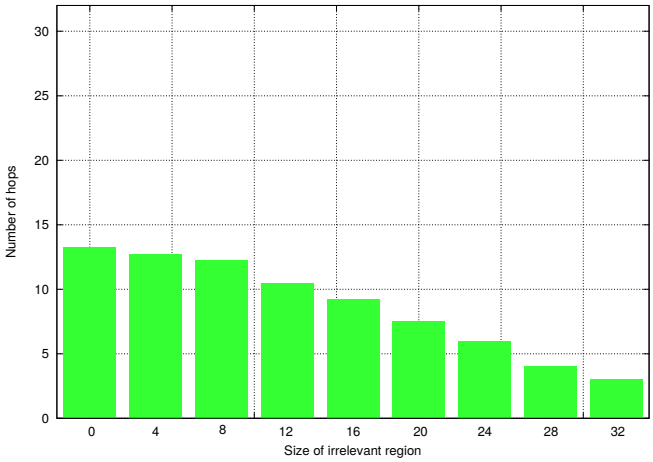

Figure 14. The effect of the size of the irrelevant step region

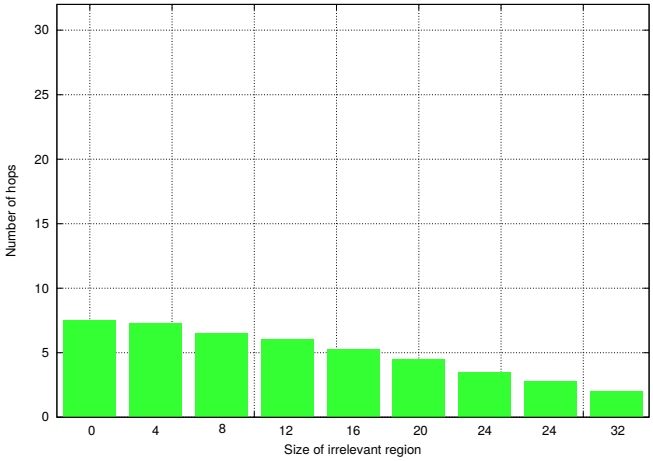

Figure 15. The effect of the size of the irrelevant step region

our system mostly the case. The optimization proposed here, will reduce the amount of AVG traffic and therefore alleviates the problem described in [7].

The optimization used and described in this paper was developed with the grid-based agile manufacturing environment in mind. The same approach can also be used in situations where workers are available, who can do one or more specific tasks needed in a certain project and where cooperation is required to reach the final goal of the project.

VI. Related work

This section will start with an overview on agent-based manufacturing. Especially the planning part will be given attention. Next, related problems in production planning, operations research and optimisation will be presented.

Important work in field of agent-based manufacturing has already been done. Paolucci and Sacile [8] give an extensive overview of what has been done. Their work focuses on simulation as well as production scheduling and control [9]. The main purpose to use agents in [8] is agile production and making complex production tasks possible by using a multiagent system. Agents are also proposed to deliver a flexible and scalable alternative for manufacturing execution systems (MES) [10] for small production companies. The roles of the agents in this overview are quite diverse. In simulations agents play the role of active entities in the production. In production scheduling and control agents support or replace human operators. Agent technology is used in parts or subsystems of the manufacturing process. The planning is mostly based on the type of planning that is used in MES. This type of planning is normally based on batch production. We based the manufacturing process as a whole on agent technology. In our case, a co-design of hardware and software was the basis. The planning will be done on a single product basis and not on batch production.

Bussmann and Jennings [11][12] used an approach that compares in some aspects to our approach. The system they describe introduced three types of agents, a workpiece agent, a machine agent and a switch agent. Some characteristics of their solutions are:

- The production system is a production line that is built for a certain product. This design is based on redundant production machinery and focuses on production availability and a minimum of downtime in the production process. Our system is a grid and is capable to produce many different products in parallel;
- The roles of the agents in this approach are different from our approach. The workpiece agent sends an invitation to bid for its current task to all machine agents. The machine agents issue bids to the workpiece agent. The workpiece agent chooses the best bid or tries again. This is what is known as the contract net protocol. In our system the negotiating is between the product agents, thus not disrupting the machine agents;
- They use a special infrastructure for the logistic subsystem, controlled by so called switch agents. Even though the practical implementation is akin to their solution, in our solution the service offered by the logistic subsystems can be considered as production steps offered by an equiplet and should be based on a more flexible transport mechanism.

So there are however important differences to our approach. The solution presented by Bussmann and Jennings has the characteristics of a production pipeline and is very useful as such, however it is not meant to be an agile multi-parallel production system as presented here. Their system uses redundancy to overcome the the problem that arises in pipeline-based production when one of the production systems fails or becomes unavailable. The planning is based on batch processing.

Other authors focus on using agent technology as a solution to a specific problem in a production environment. In [13] a multi-agent monitoring is presented. This work focusses on monitoring a manufacturing plant. The approach we use monitors the production of every single product. The work of Xiang

and Lee [14] presents a scheduling multiagent-based solution using swarm intelligence. This work uses negotiating between job-agents and machine-agents for equal distribution of tasks among machines. The implementation and a simulation of the performance is discussed. We did not focus on a specific part of the production but we developed a complete production paradigm based on agent technology in combination with a production grid. This model is based on two types of agents and focuses on agile multiparallel production. The role of the product agent is much more important than in the other agent-based solutions discussed here. In our model, the product agent can also play an important role in the life-cycle of the product [15]. The design and implementation of the production platforms and the idea to build a production grid can be found in Puik [16].

The problem presented here is also related to the famous travelling salesman problem (TSP) [17]. However, there are important differences. In our case we try to reduce the number of hops, not to find the shortest path. This has to do with the fact that in our manufacturing model the transport of products in the grid should be reduced to prevent the situation of congestion and to reduce the production time for a certain product. In our system the order of production machines to be visited by a certain product is sometimes fixed and sometimes irrelevant, making it a problem that differs from TSP.

A transport related problem in manufacturing is known as the job shop scheduling problem [18]. This problem plays an important role in standard production cell-based systems. In that case the production time and availability of production cells is the basis for the problem to be solved.

VII. Conclusion

In this paper, a path planning optimization approach has been proposed an tested. The optimization turned to work out as expected and results in a reduce of traffic among the production machines. The optimization might be useful in other situations as well, especially in situations of production systems where the transport becomes a bottleneck. In future research, other step classes can be included like the situation where the order of sequences (tuples) of steps is irrelevant.

References

[1] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Optimizing product paths in a production grid," The Fourth International Conference on Intelligent Systems and Applications (Intelli 2015), St. Julians, Malta, 2015, pp. 150–156.

[2] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Implementation of manufacturing as a service: A pull-driven agent-based manufacturing grid," Proceedings of the 11th International Conference on ICT in Education, Research and Industrial Applications (ICTERI 2015), Lviv, Ukraine, 2015, pp. 172–187.

[3] M. Wooldridge, An Introduction to MultiAgent Systems, Second Edition. Sussex, UK: Wiley, 2009.

[4] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Decentralized autonomous-agent-based infrastructure for agile multiparallel manufacturing," Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2011) Kobe, Japan, 2011, pp. 281–288.

[5] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang, "Reconfigurable manufacturing systems: the state of the art," International Journal of Production Research, vol. 46, no. 4, 2008, pp. 599–620.

[6] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Production scheduling in an agile agent-based production grid," Proceedings of the Intelligent Agent Technology (IAT 2012), Macau, 2012, pp. 293–298.

[7] L. v. Moergestel, J.-J. Meyer, E. Puik, D. Telgen, M. Kuijl, B. Alblas, and J. Koelewijn, "A simulation model for transport in a grid-based manufacturing system," The Third International Conference on Intelligent Systems and Applications (Intelli 2014), Seville, Spain, 2014, pp. 1–7.

[8] M. Paolucci and R. Sacile, Agent-based manufacturing and control systems : new agile manufacturing solutions for achieving peak performance. Boca Raton, Fla.: CRC Press, 2005.

[9] E. Montaldo, R. Sacile, M. Coccoli, M. Paolucci, and A. Boccalatte, "Agent-based enhanced workflow in manufacturing information systems: the makeit approach," J. Computing Inf. Technol., vol. 10, no. 4, 2002, pp. 303–316.

[10] J. Kletti, Manufacturing Execution System - MES. Berlin Heidelberg: Springer-Verlag, 2007.

[11] S. Bussmann, N. Jennings, and M. Wooldridge, Multiagent Systems for Manufacturing Control. Berlin Heidelberg: Springer-Verlag, 2004.

[12] N. Jennings and S. Bussmann, "Agent-based control system," IEEE Control Systems Magazine, vol. 23, no. 3, 2003, pp. 61–74.

[13] D. Ouelhadj, C. Hanachi, and B. Bouzouia, "Multi-agent architecture for distributed monitoring in flexible manufacturing systems (fms)," International Conference on Robotics and Automation, ICRA, 2000, pp. 2416–2421.

[14] W. Xiang and H. Lee, "Ant colony intelligence in multi-agent dynamic manafacturing scheduling," Engineering Applications of Artificial Intelligence, vol. 16, no. 4, 2008, pp. 335–348.

[15] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Embedded autonomous agents in products supporting repair and recycling," Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2013) Mexico City, 2013, pp. 67–74.

[16] E. Puik and L. v. Moergestel, "Agile multi-parallel micro manufacturing using a grid of equiplets," Proceedings of the International Precision Assembly Seminar (IPAS 2010), 2010, pp. 271–282.

[17] W. Cook, In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, 2012.

[18] S. Mirshekarian and D. Sormaz, "Correlation of job-shop scheduling problem features with scheduling efficiency," Expert Systems with Applications, vol. 62, 2016, p. 131147.