Patching process optimization in an agent-controlled timber mill

authored by **Matthias Wolfgang Hofmair, Martin Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi**

and published in *Journal of Intelligent Manufacturing*.

Automation and Control Institute (ACIN)
TU Wien
Gusshausstrasse 27-29/E376
1040 Vienna, Austria

Internet: www.acin.tuwien.ac.at
E-mail: office@acin.tuwien.ac.at
Phone: +43 1 58801 37601
Fax: +43 1 58801 37699

Patching Process Optimization in an Agent-controlled Timber Mill

Matthias Hofmair · Martin Melik-Merkumians · Martin Böck · Munir Merdan · Georg Schitter · Andreas Kugi

Received: date / Accepted: date

Abstract Repair and patching of wood defects is a costly process of inline production in timber industry. A large variety of plain as well as laminated wooden products demands for offline human interaction and skilled handcrafting in order to achieve the desired quality of the final products. The EU FP7 project Hol-I-Wood PR demonstrates the transformation of a traditional wood patching line for shuttering panels into a fully automated, flexible patching plant.

The focus of this paper is set on the optimization of the different production steps of a patching robot, which comprises optimal patch placement, path planning and trajectory generation. Based on this, the processing time of each workpiece can be accurately estimated. These computations serve as an input for advanced panel scheduling, which assigns panels to one of several identical parallel patching lines in a throughput-optimal manner. In order to ensure high modularity of the components and scalability for various wood mills, an agent-based approach was chosen for the implementation of the automation system.

Matthias Hofmair
Automation and Control Institute
Vienna University of Technology
Gußhausstraße 27-29 / E376
A-1040 Vienna, Austria
Tel.: +43 (0)1 58801-376269
Fax: +43 (0)1 58801-37699
E-mail: hofmair@acin.tuwien.ac.at

Martin Melik-Merkumians
Automation and Control Institute
Vienna University of Technology
Gußhausstraße 27-29 / E376
A-1040 Vienna, Austria
Tel.: +43 (0)1 58801-37688
Fax: +43 (0)1 58801-37699
E-mail: melik-merkumians@acin.tuwien.ac.at

Fig. 1 Unprocessed shuttering panel.

Keywords Wood Patching Robot · Patch Placement · Polygon Covering · Path Planning · Traveling Salesman Problem · Trajectory Generation · Agent Technology

1 Introduction

Repair of wood defects is the most time-consuming and disruptive process of inline production in timber industry. Major artifacts determining the quality of surfaces are resin galls and loose dead knots, a sample of which is shown in Figure 1. Patching is necessary to ensure homogenous material quality of laminated wood products. This process still requires a lot of manual labor.

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

Fig. 2 Defect covered with three patches.

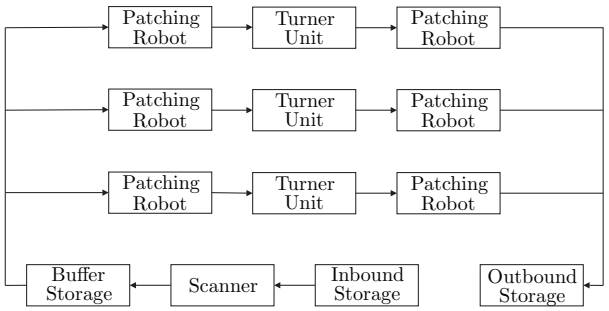

Fig. 3 Flowchart of the patching plant.

Only in recent years, industry has taken up the matter of automated patching of wood defects. Given the contour and location of the defects, basically two approaches do exist in industry: First, the defect is milled by numerically controlled machines and either an individually shaped dowel or putty is used to seal the hole. This process is rather expensive, but in turn offers the possibility to correct the wood without impairing its appearance. Second, small defects and cracks can be covered using putty only. After the putty is applied by an extremely fast manipulator, the panels are exposed to UV-light which makes the putty cure.

However to the authors' knowledge, none of these solutions nor the task of wood patching itself have been thoroughly investigated in a scientific context. The innovation in this area is mostly industrially driven. Nevertheless, there is vast potential for process optimization that requires rigorous scientific methods, even more so with the patching technique presented in this paper.

In the patching process under consideration, the defect is eliminated by drilling the respective area and then inserting a patch with high pressure to seal the hole, see Figure 2. No glue or putty is used. Since the patches have a fixed diameter of 30mm, big defects require several patches to be covered entirely. This technique is employed by semi-automatic patching tools, which are the current industrial standard[1] for patching shuttering panels. Compared to a CNC-machine, they are inexpensive and robust and shall therefore be integrated into the new automated production plant.

The patching tools are operated by a worker carrying out two major tasks: defect classification and localization, as well as visual positioning of the rectangular shuttering panel, such that the defect is exactly located beneath the patching tool. Considering the fact that defect classification and localization require much less time than patching, the envisaged plant layout consists of one scanning line and three parallel patching lines, as shown in Figure 3. Each patching line incorporates two patching robots and one turner unit.

The panels are taken from the inbound storage and scanned for defects. The scanner's core[2] is a timely synchronized camera network. The acquired pictures are merged into one for the bottom- and one for the top-side. They serve as input for defect-detection and -classification algorithms, which describe the border of the defects by polygons. The time-synchronization enables to scan the panels while in motion. After that the panels go to the buffer storage, which consists of several panel stacks. Besides its obvious buffer function, it also provides the option to choose the panel to be processed next. From there the panels are assigned to one of the three patching lines where they are patched on both sides and then collected in the outbound storage.

This paper presents a comprehensive concept for the throughput-optimal automation of the wood patching plant for shuttering panels described above. Thereby, a major task is concerned with processing time optimization and estimation for each individual panel. However, processing time optimization of individual workpieces is a very problem-specific matter, which is why the literature on this topic is quite diverse.

For instance, Mucientes et al. (2008) deals with a complex production scenario in the wood furniture industry. The processing time estimate of each workpiece is based on polynomials with multiple inputs, such as its dimension and material quality. A set of fuzzy rules is used to determine which product category, i.e. which polynomial, resembles the current product the most. Moreover, an evolutionary algorithm is employed to improve this set of fuzzy rules.

In Or and Duman (1996), the automated production process of printed circuit boards (PCB) is described. This process is split up into three optimization problems which have to be solved for each type of PCB.

[1] Approximately 1500 of these patching tools are in use in Europe.

[2] For details refer to our partner company MiCROTEC, see http://www.microtec.eu/en.

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

The process sequencing problem is a classical traveling salesman problem, the assignment of components to feeder cells of the placement machines is solved as a quadratic assignment problem and, in case the production of one PCB is managed by several placement machines, a load balancing problem has to be considered. Heuristic solution procedures provide a reliable basis for processing time estimation.

In Rubinovitz and Wysk (1988), a CAD model of the workpiece and a model of the robot's motion capabilities are used to determine the optimal processing sequence of weld seams. First, the robot trajectories between each pair of weld positions are generated. The travel times for all point-to-point movements are then used to formulate a traveling salesman problem for the purpose of defining the time-optimal processing sequence. These computations are a useful basis for processing time estimation of the workpiece.

An algorithm for online generation of painting trajectories for spraying robots can be found in Vincze et al. (2004). Laser range sensors at the robot end-effector are used to determine the geometry of the workpiece. The main idea is to split the geometry into basic surfaces for which generic paint trajectories are saved in a procedure library. Assuming a large scale spraying plant that consists of a dedicated scanning unit and several spraying stations, these trajectories enable processing time estimation for subsequent process scheduling.

Based on the processing time estimates for each panel, a further task is optimal assignment of panels to the three parallel patching lines. This problem presents itself as classical *parallel machine scheduling with a single server*. Its complexity is analyzed in Hall et al. (2000) and in Hall et al. (2000); Kravchenko and Werner (1997); Kim and Lee (2012) algorithms are presented to solve this problem in polynomial time. State-of-the-art surveys on assembly line balancing and scheduling using evolutionary algorithms are presented in Tasan and Tunali (2008); Gen and Lin (2013). In Barbati et al. (2012); Kouiss et al. (1997), Multi-Agent Technology (MAT) is applied to dynamic scheduling problems.

In this application, a plant control based on MAT is strived for, because of manifold reasons: The plant components can be developed with little to none dependencies. This provides plant owners with a best-of-breed approach for their plant components without increased costs for interoperability. By defining simple interfaces, which represent the tasks carried out by the plant components, the overall plant control is independent of the specific plant components in use. Therefore, components can easily be replaced by functionally equivalent components of other brands or types, as shown in Vrba and Mařík (2010); Bussmann et al. (2004); Leitão (2009). A further advantageous feature of this approach is the simplified integration of legacy equipment in modern plant control. This is of particular importance in the wood industry in Europe, as there are many existing wood mills with aged control technology, which needs to be refurbished in the years to come. The agent-based approach helps to ensure a smooth transition from the currently used control equipment to new technologies. A further benefit of this approach is the obtained scalability of the plant. The agent-based system can easily cope with one or more parallel patching lines, without agent-code modification. Also the agents' ability to react to changes in the system, like plant faults, constitutes an important feature in the harsh production environment of the wood industry, see Pěchouček and Mařík (2008).

The paper is structured as follows: The basics of the Multi-Agent Architecture for the patching plant under consideration are briefly outlined in Section 2. Section 3 describes optimal patch placement, path planning and trajectory generation of the patching robot. Based on this, a reliable estimate of the processing time of an individual panel is given. Section 4 presents simulation results for the optimization algorithms and Section 5 contains conclusions and gives an outlook on future work.

The novelty of this paper is the successful combination of one-of-a-kind production with throughput maximization. Usually, one-of-a-kind production only refers to extremely complex products that are produced on demand such as ships, planes or industrial plants. This is due to the fact that wood is a natural product and consequently each workpiece is unique. Thus, an optimization must be carried out for each individual shuttering panel. On average, an unprocessed shuttering panel exhibits six defects per square meter per panel side, ten percent of which require more than one patch to be covered entirely. The panels are scanned for defects at a rate of approximately 0.2Hz. So there is a time slot of 5s to accomplish the entire planning task. The results of the planning algorithms are further used to accurately estimate the processing time of each panel, which serves as the basis for subsequent panel scheduling. The proposed algorithms incorporate techniques from algorithmic geometry, combinatorial optimization and time-optimal trajectory generation.

2 Multi-Agent Architecture

In order to provide the necessary flexibility for refurbishing existing wood mills and in view of the desired modularization of plant components and scalability for

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

various wood mill sizes, an agent-based approach was chosen. In contrast to more traditional methods (e.g., object-oriented design, remote procedure calls, etc.), Multi-Agent Systems facilitate decoupling of plant components. This is achieved by the generally higher abstraction level of the agent-based approach compared to purely object-oriented paradigms, as the behavior of agents more closely resembles human behavior in a simplified form, as shown in Zambonelli et al. (2000), meaning that an agent represents a goal-driven, autonomous and proactive entity. Ultimately, an agent incorporates not only the behaviors for executing a control algorithm but is also empowered to decide whether to execute the algorithm or not, which is a significant difference to a passive object and delivers a truly self-contained entity, see Jennings and Sycara (1998). In this context, each agent is only responsible for its own task/component and publishes only those services which are provided to the rest of the system. Therefore, the individual agents do not rely on other agents to perform their specific tasks. In this regard an agent-based system is similar to service-oriented control approaches as presented in Melik-Merkumians et al. (2012); Ollinger et al. (2013); Huhns et al. (2005); Ribeiro et al. (2008). But in contrast to agent-based systems, service-oriented components have less individual responsibilities (e.g., system reconfiguration in case of a fault).

There are two kinds of agents in the system:

- Functional Agents(FAs) — these agents have no direct representation in the plant and are responsible for plant wide tasks. All FAs are located in the Management Layer.
- Automation Agents(AAs) — these agents are in direct control of plant components. AAs are always part of the Automation Layer.

The agent system structure, as shown in Figure 4, is organized in three layers. The first layer, the Management Layer, encompasses the agents/agendas which are necessary for the overall plant control (see Merdan et al. (2008); Shen et al. (2006); Vrba and Mařík (2010)). This includes the plant's knowledge base, the Wood Knowledge Repository (WKR), which is accessed via the Wood Knowledge Repository Agent (WKRA), the Patching Process Agent (PPA), the Scheduling Agent (SA), the Service Directory Agent (SDA) and the Fault Diagnosis Agent (FDA).

- The WKRA manages the access to the WKR, where all essential plant data is stored. Via this agent the other agents can inform themselves about various aspects of the plant e.g., amount and location of defects on a specific panel, or the current state of components. Also the WKRA is responsible for adding new panels to the WKR as they are scanned.
- The PPA calculates the optimal processing routine, i.e. patch locations, processing sequence and robot paths. To do so, the PPA asks the WKRA for the defect locations of a panel, carries out its computations (as described in Section 3) and returns the processing routine to the WKRA along with an estimate of the processing times for each side of the panel.
- The SA collects the estimated processing times of the currently available, not processed panels and calculates, based on the current state, the optimal work schedule for the available patching lines.
- The SDA is a lookup table where all other agents register themselves at start-up and publish their services. It informs agents about where requested services can be found (e.g., at start-up the PPA does not know the actual address of the WKRA). If an agent is not able to perform its assigned task it renounces itself from the SDA. In this way, a dynamic handover of responsibilities is achieved. By that the system can easily continue its service in a gracefully degraded state.
- If an agent experiences unusual difficulties or system reactions, e.g., a panel gets stuck in a patching robot causing that the panel never leaves the robot, the responsible and/or affected agents inform the FDA about their actual state and request analysis. This analysis may also affect other agents which are not directly involved (e.g., the SA has to reschedule after a patching line is taken offline by the FDA). The FDA steadily observes plant behavior, comparing it to the expected behavior in order to detect and avoid global errors (e.g., if the feeder conveyor to the patching lines gets stuck all three patching lines will seem to be stuck).

The second layer, the Integration Layer, is the intermediate communication layer between the agents on the Automation Layer and the agents on the Management Layer. It provides communication interfaces on the field level for direct device control and it ensures compatibility to the (control) hardware executing the agents.

The third layer, the Automation Layer, comprises all AAs, which control the physical components of the plant. Each AA consists of two parts. The Proactive Control (PC)[3] is responsible for the optimization algo-

[3] The open-source platform Java Agent DEvelopment framework (JADE) (Jade 2014) is used for development and execution of the PC. When using the JADE framework, the task of the SDA can be performed by JADE's Directory Facilitator agent.

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

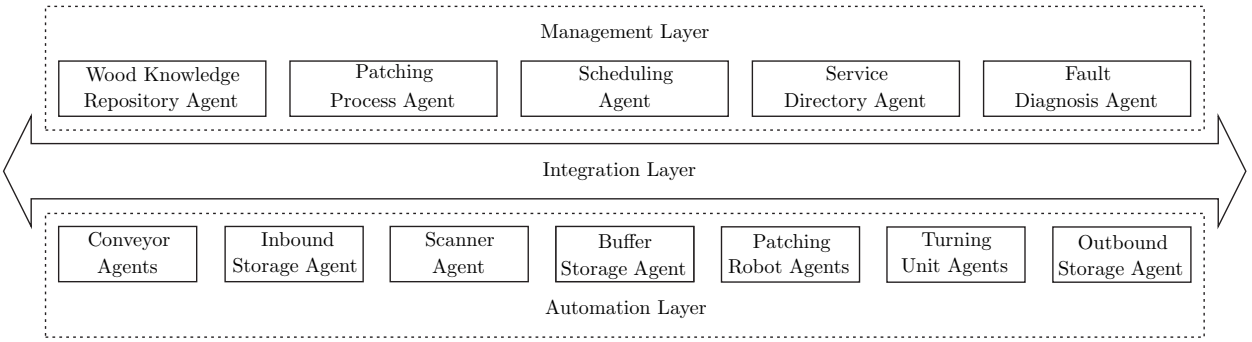

Fig. 4 Multi-Agent Architecture of the patching plant.

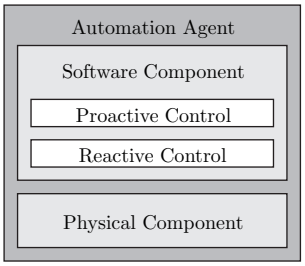

Fig. 5 Architecture of a generic AA.

rithms and communication to other agents. It processes the information from the agent system and issues orders to the Reactive Control (RC). The RC is the real-time capable control software of the plant components, which performs its tasks based on the parameters given by the PC. Process information acquired via the process interface (e.g., I/O values) is forwarded from the RC to the PC via a control software specific interface. In case of this implementation, the RC was realized as an IEC 61499 program. Process data needed by the PC is forwarded via so-called service interface function blocks, whose task it is to connect the control environment with system parts (e.g., the agent system) which are not part of the control environment, see Hegny et al. (2008). This architecture of a generic AA is depicted in Figure 5, further details can be found in Lepuschitz et al. (2011). In case of a fault, the RC is also responsible for putting the component into a safe state, see Vallée et al. (2009).

For the considered plant, the following AAs are defined:

- Each Conveyor Agent (ConA) manages its assigned section of the transport grid of the plant. It is responsible for detecting local errors on its section (e.g., a stuck panel) and for reporting these errors to the SA and FDA.
- The Inbound Storage Agent (ISA) manages the infeed of panels to the plant.
- The Scanner Agent (ScA) uses its camera system to capture optical information on the panel, thus detecting the defects' location and shape. The panel is assigned an ID, then this information, called defect list, is sent to the WKRA.
- The Buffer Storage Agent (BSA) receives instructions from the SA and accordingly feeds the panels into the assigned patching line. It informs the Patching Robot Agents(PRAs) and the SA about its actions.
- The PRA requests the configuration parameters and the processing routine from the WKRA and provides this information to the patching robot.
- The Turning Unit Agents(TAs) are in control of the turner units, where the panels are flipped upside down.
- The Outbound Storage Agent (OSA) manages the outbound storage and the outgoing flow of the panels.

The typical workflow for a wooden panel is as follows:

1. The panel enters the plant via the inbound storage. The ISA announces a new panel to the system.
2. The panel gets scanned and the ScA sends the defect list to the WKRA and requests the data to be inserted into the WKR.
3. The WKRA sends the defect list of the panel to the PPA and asks for the optimal processing routine and estimated processing time.
4. Based on the current workload and the estimated processing time, the SA assigns the panel to a patching line.
5. As the panel arrives at the patching robot, the PRA requests the trajectory from the WKRA and enters it into the RC of the patching robot.
6. After the patching process is completed, the patched panel leaves the system via the outbound storage. The OSA informs the WKRA that the panel was processed successfully.

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

The presented system architecture automates the generation and execution of the the complete patching process.

3 Optimizing the Patching Process and Processing Time Estimation

For both sides of each panel, the scanner provides a list of defects. This defect list serves as an input for the following algorithms of the PPA:

1. Each defect needs to be entirely covered by a minimum number of patches.
2. The minimum cost path, i.e. processing sequence, is calculated with respect to a desired cost function.
3. The geometric robot path is parametrized in the time t in order to yield (nearly) time-optimal trajectories for the robot.
4. Based on the results of items 1-3 the processing time of each panel side is estimated.

The tasks listed above are described in the following subsections.

3.1 Patch Placement

Due to manufacturing reasons there is only one shape of patches, namely cylinders with a radius of $r = 0.015$m. They allow to patch approximately 85% of all wood defects using merely one patch. The other 15% are too big to be covered by one patch only. Since no glue is used, it is obvious that the more patches are placed next to each other, the more fragile this whole arrangement becomes, see Figure 2. Some defects are even too big for patching and thus the associated panel must be rejected by the PPA.

Therefore, a patch placement algorithm has to compute the minimum number of patches and their arrangement necessary to cover these big defects. Striving for a minimum number of patches serves two important purposes. First, each patch requires approximately $T_p = 2$s processing time, the positioning time not included. Second, since the allowed maximum number of patches per defect is limited due to quality reasons, minimizing the number of patches indirectly but nonetheless significantly contributes to waste reduction.

The input for the patch placement algorithm is the defect list $\mathbf{\Delta} = \{\mathbf{D}_i\}$, $i = 1, \ldots, N$, of the respective panel side. Each element of the defect list defines one closed defect polygon by a list of vertices, $\mathbf{D}_i = \{\mathbf{d}_{i,j}\}$, $\mathbf{d}_{i,j} = \begin{bmatrix} x_{i,j} & y_{i,j} \end{bmatrix}^{\mathrm{T}}$, $j = 1, \ldots, n_{\mathbf{D}_i}$, where $x_{i,j}$ and $y_{i,j}$ denote the coordinates of vertex j of defect i with respect to the front left corner of the panel. It describes the contour of the defect by connecting vertex $\mathbf{d}_{i,j}$ to vertex $\mathbf{d}_{i,j+1}$ until finally the last vertex $\mathbf{d}_{i,n_{\mathbf{D}_i}}$ is again connected to the first vertex $\mathbf{d}_{i,1}$.

The polygon $\mathbf{D}_i$ has to be entirely covered by a number $n_{\mathbf{P}_i}$ of circles of a given radius r. Each such circle is described by its center coordinates $\mathbf{p}_{i,k} = \begin{bmatrix} x_{i,k} & y_{i,k} \end{bmatrix}^{\mathrm{T}}$. All patches required to cover the defect $\mathbf{D}_i$ are merged into the set $\mathbf{P}_i = \{\mathbf{p}_{i,k}\}$, $k = 1, \ldots, n_{\mathbf{P}_i}$.

The task is now to determine the minimum number $n_{\mathbf{P}_i}$ of patches and their arrangement to cover the entire defect. Furthermore, a set of constraints ensuring that the patches adhere to the panel (and do not fall out in later production stages) has to be met. In particular, each patch must have a minimum overlap with solid wood and the patches themselves must not overlap too much. This is formulated as an optimization problem

$$\min_{n_{\mathbf{P}_i}, \mathbf{P}_i} \; n_{\mathbf{P}_i} \tag{1a}$$

$$\tilde{\boldsymbol{\gamma}}(\mathbf{D}_i, \mathbf{P}_i, n_{\mathbf{P}_i}) \leq \mathbf{0}, \tag{1b}$$

where $\tilde{\boldsymbol{\gamma}}$ is the vector of constraints. In order to check the defect covering and the overlap of each patch with solid wood, the defect polygon is intersected with each patch circle. Also the patch circles themselves are intersected with each other to ensure sufficiently small overlap of the patches.

The problem with the above formulation is that the number of optimization variables $\tilde{n}_o = 1 + 2n_{\mathbf{P}_i}$ increases with the number of patches $n_{\mathbf{P}_i}$. Each patch has to be placed individually taking into account all the other $n_{\mathbf{P}_i} - 1$ patches. Furthermore, (1) constitutes a mixed-integer optimization problem, which is known to be quite challenging.

To overcome this issue, the whole patch placement algorithm is founded on the idea of hexagonally closest packaging, see Williams (1979) and Figure 6. A hexagon is the one polygon that comes closest to the circular shape, while at the same time it can still be arranged in a pattern fully covering a plane, leaving no holes in between the hexagons. Thus minimal overlap between the circles is ensured. This naturally implies maximum covered area (for a fixed number of patches) and consequently guarantees that the minimum number of patches is utilized.

The algorithm simply starts by creating a hexagon tessellation in a sufficiently large part of the xy-plane, where each hexagon of side length r represents a circle of equal radius. The defect polygon is put in the middle of the tessellated area, i.e. the centroid of the defect polygon coincides with the centroid of the tesselated area. The optimization is now achieved by moving the defect along the x- and y-axis, denoted by δ_x and δ_y,

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

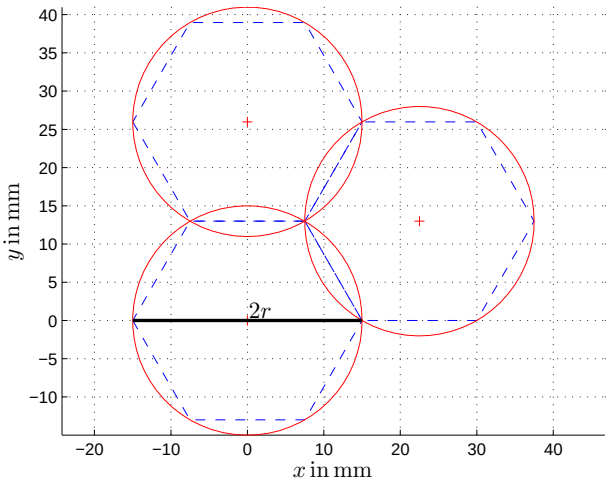

Fig. 6 Hexagonally closest packaging.

and by rotating it by an angle δ_φ until a global minimum in the sense of the previously described criterion is found.

During this process hexagons that do not intersect with the defect polygon are taken out. Others are added where the defect polygon is not covered. Compared to the original problem formulation, the patches are perfectly arranged to each other. Therefore, the task of checking the arrangement of every patch relative to all the other patches needs not be performed.

Thus the optimization problem can be implemented more efficiently in the form

$$\min_{\boldsymbol{\delta}} n_{\mathbf{P}_i} \tag{2a}$$

$$\boldsymbol{\gamma}(\mathbf{D}_i, \boldsymbol{\delta}) \leq \mathbf{0}, \tag{2b}$$

where $\boldsymbol{\delta} = \begin{bmatrix}\delta_x & \delta_y & \delta_\varphi\end{bmatrix}^{\mathrm{T}}$ is the displacement of the defect polygon relative to the tessellated area.

This way the number of optimization variables is reduced to $n_o = 3$. Moreover, the vector of constraints $\boldsymbol{\gamma}$ is significantly simplified, since the patch placement relative to each other is fixed. The majority of intersections, i.e. each patch with every other patch, becomes obsolete.

The hexagon tessellation has a periodic pattern. If the defect is moved over an x-distance of the width of a hexagon, $\bar{\delta}_x = 2r$, if it is moved over a y-distance of the height of a hexagon, $\bar{\delta}_y = \sqrt{3}\,r$, or if it is rotated around $\bar{\delta}_\varphi = \pi/3$, the pattern repeats itself. So the space of all possible solutions $\mathbb{D} = [0, \bar{\delta}_x] \times [0, \bar{\delta}_y] \times [0, \bar{\delta}_\varphi]$ is closed.

Nevertheless, the solution space still contains an infinite number of points. Only by applying a fixed step size $\boldsymbol{\delta}_d = \begin{bmatrix}\delta_{dx} & \delta_{dy} & \delta_{d\varphi}\end{bmatrix}^{\mathrm{T}}$ for the displacement of the defect, one obtains a finite number of possible solutions $N_{\mathbb{D}} = \left(\bar{\delta}_x/\delta_{dx}\right)\ \left(\bar{\delta}_y/\delta_{dy}\right)\ \left(\bar{\delta}_\varphi/\delta_{d\varphi}\right)$. A natural choice for this step size is the positioning accuracy of the patching robot. By going through all these solutions, it is guaranteed that the global minimum with respect to the positioning accuracy is obtained in a finite number of $N_{\mathbb{D}}$ steps.

3.2 Path Planning

The result of the patch placement algorithm is a patch list $\boldsymbol{\Pi} = \{\mathbf{P}_i\}$, $i = 1, \ldots, N$, i.e. a list of patch locations $\mathbf{p}_{i,k}$ the robot must approach. To keep the notation general, the patch locations are referred to as nodes $\mathbf{x}_i = \begin{bmatrix}x_i & y_i\end{bmatrix}^{\mathrm{T}}$ and merged into a node list $\mathbf{X} = \{\mathbf{x}_i\}$, $i = 1, \ldots, n$, $n = \sum_{k=1}^{N} n_{\mathbf{P}_i}$.

The path planning algorithm aims at computing the minimum cost path between a number n of nodes such that each node is visited exactly once. The costs can be chosen freely and are for instance path length, travel time or energy consumption.

This problem is very similar to the well-known traveling salesman problem, where a salesman tries to find a minimum cost round trip through a given number of cities. It is an NP-hard combinatorial optimization problem, for which a variety of different approaches are proposed in the literature. Basically exact algorithms, such as formulating the traveling salesman problem as a linear integer program, as shown in Applegate et al. (2006), and heuristic algorithms, such as Genetic (see Eiben and Smith (2003)) or Ant Colony Algorithms (see Dorigo and Stützle (2004)), are distinguished. High-performance heuristic algorithms frequently make use of Local Search Strategies, see Lin (1965), to further improve their solution.

However, the panel is not supposed to go on a round trip and come back to where it started, but to eventually move forward in the production line. Nevertheless, the known algorithms for the traveling salesman problem can still be applied.

Cost Matrix In view of maximizing productivity, the goal is to determine the time-optimal path. Therefore, the costs represent the time it takes the patching robot to move from node $\mathbf{x}_i$ to node $\mathbf{x}_j$, given by a function $c(\mathbf{x}_i, \mathbf{x}_j)$. As is frequently the case in xy-positioning, the x- and y-movement of the patching robot are assumed to be independent. Thus the costs are defined as

$$c_{i,j} = c(\mathbf{x}_i, \mathbf{x}_j) = \max(\bar{T}_{x_{i,j}}, \bar{T}_{y_{i,j}}), \tag{3}$$

$\bar{T}_{x_{i,j}}$ and $\bar{T}_{y_{i,j}}$ being the travel times in longitudinal and lateral direction, respectively. They are computed between each pair of nodes yielding the cost matrix $\mathbf{C} =$

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

$[c_{i,j}] \in \mathbb{R}^{n\times n}$. The diagonal elements of this symmetric matrix are zero, i.e. $c_{i,i} = 0$, $i = 1, \ldots, n$.

Solution In the following, two solutions to this problem are proposed: First, an Ant Colony Algorithm which is recognized as an efficient approach to tackle large problem instances of the general traveling salesman problem is described. Second, a Local Search Algorithm combined with a Receding Horizon Concept presents a problem-tailored and simple solution to the specific traveling salesman problem at hand.

For this, let us consider a set of nodes $\mathbf{X}$ that is fully connected by a set of arcs Ξ. Thus the pair $G = (\mathbf{X}, \Xi)$ is a complete graph, i.e. every node $\mathbf{x}_i \in \mathbf{X}$ is directly connected to every other node $\mathbf{x}_j \in \mathbf{X}$. Each arc $\xi_{i,j} \in \Xi$ is now associated with its respective cost $c_{i,j}$. The path planning problem is the problem of finding a minimum cost path on G from a previously specified start node, the one with the lowest x-coordinate, to an end node, the one with the highest x-coordinate, visiting all the other nodes exactly once. Therefore, the path vector can be defined as

$$\boldsymbol{\psi} = \left[\psi_1 \ldots \psi_i \ldots \psi_n\right], \; i = 1, \ldots, n \tag{4a}$$

$$\psi_i \in \{1, \ldots, n\} \setminus \{\psi_0, \ldots, \psi_{i-1}\}, \; \psi_0 = \{\}. \tag{4b}$$

The path cost is the sum of the arc costs of the respective path through the graph,

$$J = \sum_{i=1}^{n-1} c_{\psi_i,\psi_{i+1}} = \sum_{i=1}^{n-1} c(\mathbf{x}_{\psi_i}, \mathbf{x}_{\psi_{i+1}}). \tag{5}$$

3.2.1 Ant Colony Algorithm (ACOA)

The objective function (5) can be minimized by means of an Ant Colony Algorithm. The foraging behavior of ants is determined by pheromone trails they deposit on their way between food and nest. Ants tend to follow these trails, while at the same time reinforcing them. Thus, the pheromone concentration serves as the ant colony's collective memory, which develops over time. The desirability of various paths is remembered and visible for other ants. For more detailed information, the reader is referred to, e.g., Dorigo and Di Caro (1999); Dorigo and Stützle (2004). In the following, only the core functionality of all Ant Colony Algorithms, i.e. the iterative, cooperative solution construction, shall be outlined.

In every iteration $t = 1, \ldots, T$, each ant $m \in \{1, \ldots, M\}$ of a population of size M individually constructs feasible paths by making $h = 1, \ldots, n-2$ random descisions[4] based on (6). The probability that an ant m decides to move from its current node i to node j is

$$p_{i,j}^m(t) = \begin{cases} \frac{\tau_{i,j}^\alpha(t)\eta_{i,j}^\beta}{\sum_{l\in\mathcal{N}_h^m} \tau_{i,l}^\alpha(t)\eta_{i,l}^\beta} & j \in \mathcal{N}_h^m \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $\mathcal{N}_h^m$ is the so-called feasible neighborhood[5] and $\tau_{i,j}(t)$ and $\eta_{i,j} = 1/c_{i,j}$ denote the pheromone trail value and the heuristic information value on the arc $\xi_{i,j}$, respectively. The tuning parameters $\alpha \in \mathbb{N}$ and $\beta \in \mathbb{N}$ are used to scale the weight of the heuristic and the pheromone information relative to each other.

In this formulation, the heuristic value $\eta_{i,j}$ represents a priori information on the problem. The costlier an arc, the less desirable it is for an ant to use. The pheromone value $\tau_{i,j}$, on the contrary, represents learned desireablity to use that arc. Thus the pheromone matrix $\mathbf{T}(t) = [\tau_{i,j}(t)] \in \mathbb{R}^{n\times n}$, being the collective memory of all ants, changes over time as ants deposit pheromone on their paths. This is accomplished by the following pheromone update laws. First, a portion $\rho \in [0, 1]$ of pheromone evaporates

$$\tau_{i,j}(t) \leftarrow (1-\rho)\,\tau_{i,j}(t) \tag{7}$$

and second, each ant m deposits pheromone along its path

$$\tau_{\psi_i^m,\psi_{i+1}^m}(t) \leftarrow \tau_{\psi_i^m,\psi_{i+1}^m}(t) + \frac{1}{J^m(t)}, \; i = 1, \ldots, n-1, \tag{8}$$

with $\boldsymbol{\psi}^m$ and J^m according to (4) and (5), respectively. Pheromone evaporation causes "bad" paths to be forgotten, while pheromone deposition enables ants to share their experience via the collective memory.

Finally, the feasibility of the ants' paths has to be discussed. To guarantee that ants visit each node exactly once, each ant m is given a local memory

$$\mathcal{M}_h^m = \{\psi_1^m, \ldots, \psi_h^m\}. \tag{9}$$

It contains all the nodes already visited up to iteration h of the construction process and thus updates the feasible neighborhood,

$$\mathcal{N}_h^m = \{1, \ldots, n\} \setminus \mathcal{M}_h^m. \tag{10}$$

The completeness of the graph G makes each node a neighbor of every other node, i.e. each node can be reached from any other node by crossing only a single

[4] Start and end node are already fixed.

[5] Since G is a complete graph, the feasible neighborhood does not depend on the current node i but only on the previously visited nodes, see (10).

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Table 1 Parameters for the implemented algorithm were chosen experimentally. This results in similar values as proposed in the literature, e.g. Dorigo and Stützle (2004). Only for the evaporation rate ρ a very small value was chosen.

α	β	ρ	M	T	e
1	5	0.2	n	70	1

arc. Thereby it is ensured that ants cannot get stuck at a node[6].

The parameter set for the implemented Ant Colony Algorithm is given in Table 1. In order to improve its real-time capability, more importance is attached to rapid convergence compared to solution space exploration. To achieve this, the evaporation parameter ρ is set to a small value and the concept of an elitist ant is introduced, see Dorigo and Stützle (2004).

Elitist Ant The elitist ant, denoted as $(.)^{\varepsilon}$, holds the best-so-far path $\boldsymbol{\psi}^{\varepsilon}$, i.e. the best path found since the start of the algorithm. This gives the possibility to stop the algorithm any time, still yielding a useful result. Additionally, the elitist ant deposits pheromone in every iteration of the algorithm,

$$\tau_{\psi_i^{\varepsilon},\psi_{i+1}^{\varepsilon}}(t) \leftarrow \tau_{\psi_i^{\varepsilon},\psi_{i+1}^{\varepsilon}}(t) + \frac{e}{J^{\varepsilon}(t)},\ i = 1,\dots,n-1, \tag{11}$$

where $e \in \mathbb{R}_+$. With growing e, this provides increasingly strong reinforcement of the best-so-far path.

By the time the algorithm terminates, a number of $M\,T$ paths were constructed, the best of which is $\boldsymbol{\psi}^{opt} = \boldsymbol{\psi}^{\varepsilon}$. Applying the Ant Colony Algorithm to the patch list $\boldsymbol{\Pi}$ yields $\boldsymbol{\Pi}^{opt}$, which is now sorted according to the time-optimal path $\boldsymbol{\psi}^{opt}$. In the following, two more measures for speeding up the algorithm in view of the considered problem are proposed.

Patch Clustering Since the traveling salesman problem is NP-hard, it is of vital importance for the computation time of the algorithm to keep the problem size, that is the number of nodes n, at a minimum. For this specific application, some selected measures can be taken. Suppose a panel exhibits eight small and two big defects, which makes a total of $N = 10$ defects. Each of these big defects requires three patches. So there are in total $n = 14$ patches to be placed. Naturally the patches covering the big defects are very close to each other. So by clustering the patches according to their respective defect, the problem size decreases from $n = 14$ to $\bar{n} = N = 10$.

[6] If G was not complete, it might happen that the feasible neighborhood $\mathcal{N}_h^m = \{\}$ before the construction process is finished.

Thus the path planning algorithm is only applied to the reduced set $\mathbf{X} = \bar{\boldsymbol{\Pi}} = \{\mathbf{p}_{i,1}\}$, $i = 1,\dots,N$ instead of the set $\mathbf{X} = \boldsymbol{\Pi} = \{\mathbf{P}_i\}$ with $\mathbf{P}_i = \{\mathbf{p}_{i,k}\}$, $k = 1,\dots,n_{\mathbf{P}_i}$. This considerably lowers the computation time, while at the same time the path costs remain virtually the same.

Start Solution Taking into account the flow of the rectangular shuttering panels through the production line, as described in Figure 3, it is beneficial to add a from-left-to-right path and a nearest-neighbor path into the set of start solutions.

The risk with this approach is that in the early iterations of the algorithm these "good" start solutions might build up very high pheromone values due the reinforcement by the elitist ant. This results in a monodirectional search, ignoring other promising solutions. However, this can be avoided by decreasing the parameter e in (11).

3.2.2 Local Search Receding Horizon Algorithm (LSRHA)

The core of this algorithm is Local Search, which is based on the concept of λ-optimality and was initially proposed in Lin (1965), see also Helsgaun (2000, 2009). Then, the Local Search Algorithm is combined with a Receding Horizon Concept, which, for instance, is used in air traffic management, see Hu and Chen (2005b,a); Hu et al. (2007); Zhan et al. (2010). This concept was originally proposed for optimal control problems, such as model predictive control, see, e.g., Allgöwer et al. (1999).

3-optimality The concept of λ-optimality is defined as follows: A path is said to be λ-optimal, if it is impossible to obtain a tour with smaller cost by replacing any λ of its arcs $\xi_{i,j}$ by any other set of λ arcs[7]. Consequently, an n-optimal tour, i.e. $\lambda = n$, is globally optimal. So by increasing λ, the problem gets more general, in the sense that the number of possible exchanges increases. This way, one receives increasingly strong necessary conditions for optimality, but this comes at the cost of increasing computational effort.

Computer experiments showed that setting $\lambda = 3$ is the most efficient trade-off between solution quality and computational costs, see Lin (1965). By means of 3-exchanges, it is possible to realize path inversions and insertions.

[7] The implication being that a subset $\tilde{\lambda} \leq \lambda$ of the arcs may be contained in both sets and thus remain the same.

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

Local Search Algorithm The algorithm begins by generating start solutions, see Subsection 3.2.1. Each of these solutions is then refined by randomly performing 3-exchanges until no further improvement can be achieved. Details on how to efficiently implement the Local Search can be found in Lin (1965). The best one is assumed to be the optimal path $\boldsymbol{\psi}^{opt}$.

The Receding Horizon Concept is now utilized to split the entire problem instance in smaller subproblems to which the Local Search Algorithm is applied in an iterative manner.

Receding Horizon Concept It is based on the fact that a partial path is hardly ever influenced by nodes very far from it, in particular, if there are several other nodes in between. Consequently, path quality does not suffer when disregarding nodes outside the vicinity of the current local optimization horizon. However, since the traveling salesman problem is NP-hard, the computational effort is reduced drastically. Therefore, a piecewise local optimization process is proposed. It starts at the left panel end and advances on to the right end by subdividing the total problem instance of n nodes into smaller subproblems of $h_o \ll n$ nodes.

In order to start this algorithm, first the nodes are sorted in ascending x-direction, i.e. the from-left-to-right path $\boldsymbol{\psi}^{l2r} = \left[\psi_1^{l2r}, \ldots, \psi_n^{l2r}\right]$ is generated. Now, beginning at the left panel end, the Local Search Algorithm is applied to subproblems of size h_o. However, only the first $h_i < h_o$ nodes are assigned to the optimal final path. In this context, h_o and h_i are called optimization and implementation horizon, respectively.

The following procedure is applied iteratively, whereas $\Delta\boldsymbol{\psi}^{(0)} = \left[\psi_2^{l2r}, \ldots, \psi_{1+h_o}^{l2r}\right]$ qualifies as a suitable initial condition.

After iteration step k, the optimal final path is given by $\boldsymbol{\psi}^{(k),*} = \left[\psi_1^*, \ldots, \psi_{1+h_ik}^*\right]$. In the next iteration step $k+1$, the partial path defined by the nodes $\Delta\boldsymbol{\psi}^{(k+1)} = [\psi_{2+h_ik}, \ldots, \psi_{1+h_ik+h_o}]$ has to be optimized for the optimization horizon h_o, starting at node ψ_{1+h_ik}. As an initial guess of the solution of this optimization problem, the first h_i nodes of the remaining from-left-to-right path $\boldsymbol{\psi}^{l2r}$ are appended to the optimal partial path $\Delta\boldsymbol{\psi}^{(k),*}$ of the previous iteration k, i.e. $\left[\psi_{2+h_ik}^{(k),*}, \ldots, \psi_{1+h_i(k-1)+h_o}^{(k),*}, \psi_{2+h_i(k-1)+h_o}^{l2r}, \ldots, \psi_{1+h_ik+h_o}^{l2r}\right]$. Application of the Local Search Algorithm presented in Subsection 3.2.2 yields the optimal partial path

$$\Delta\boldsymbol{\psi}^{(k+1),*} = \left[\psi_{2+h_ik}^{(k+1),*}, \ldots, \psi_{1+h_ik+h_o}^{(k+1),*}\right]. \tag{12}$$

In the sense of the Receding Horizon Concept, only the first h_i nodes of (12) are added to the optimal final path such that $\boldsymbol{\psi}^{(k+1),*} = \left[\psi_1^*, \ldots, \psi_{1+h_ik}^*, \psi_{2+h_ik}^{(k+1),*}, \ldots, \psi_{1+h_i(k+1)}^{(k+1),*}\right]$. This procedure is repeated until[8] $1 + h_i(k+1) \geq n-1$. Figure 7 illustrates this process. The hatched nodes represent the partial path $\Delta\boldsymbol{\psi}^{(k+1),*}$ under consideration in iteration step $k+1$. It is of length h_o and the superscript $(.)^*$ indicates it has already been optimized. Now the first h_i nodes of $\Delta\boldsymbol{\psi}^{(k+1),*}$ are appended to the final optimal path of the previous iteration step k to form the current optimal final path $\boldsymbol{\psi}^{(k+1),*}$, i.e. the white nodes. The black nodes are the remaining nodes which have not been optimized yet.

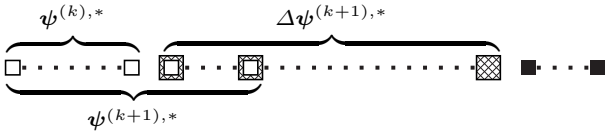

Fig. 7 Illustration of the Receding Horizon Concept.

Finally, the node list $\boldsymbol{\Pi}$ is sorted according to $\boldsymbol{\psi}^{opt} = \boldsymbol{\psi}^{(k+1),*}$ and $\boldsymbol{\Pi}^{opt}$ is handed over to the trajectory generator.

3.3 Sine-Square-Trajectory-Generator (SSTG)

Receiving two consecutive patch locations $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ of the patch list $\boldsymbol{\Pi}^{opt}$ as an input, a nearly time-optimal trajectory for the patching robot has to be planned. Naturally this trajectory generator must account for the dynamic limitations of the patching robot, namely maximum velocity v_{max}, acceleration a_{max} and jerk j_{max}.

Trajectories with high acceleration are likely to induce vibrations in the machine structure. This puts an unnecessary burden on the machine and moreover hampers precise positioning. The choice of sufficiently smooth trajectories is a measure to avoid these problems, see Biagiotti and Melchiorri (2008).

In general the trajectory generator computes the trajectory between a start point $\left[x_d(t_i)\ y_d(t_i)\right]^{\mathrm{T}} = \mathbf{x}_i$ and an end point $\left[x_d(t_j)\ y_d(t_j)\right]^{\mathrm{T}} = \mathbf{x}_j$. The travel time is denoted by $t_d = t_j - t_i$. Since the patching robot is a time-invariant dynamical system, one sets $t_i = 0$ without loss of generality. Similarly, $\mathbf{x}_i = \mathbf{0}$ and $\mathbf{x}_j \leftarrow \Delta\mathbf{x} = \mathbf{x}_j - \mathbf{x}_i$ is applied.

The dynamics of the patching robot are decoupled in x- and y-direction. Therefore, the trajectory can be planned in longitudinal and lateral direction separately.

[8] The leftmost node ψ_n^{l2r} always remains the last node, so it is not part of any optimization problem.

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

The idea is to prescribe the desired accelerati $a_d(T_j, T_a, T_v, t)$ in the form

$$a_d = \begin{cases} a_{max} \sin^2(\omega\, t) & 0 \le t \le \bar{T}_1 \\ a_{max} & \bar{T}_1 < t \le \bar{T}_2 \\ a_{max} \sin^2(\omega\,(t - T_a)) & \bar{T}_2 < t \le \bar{T}_3 \\ 0 & \bar{T}_3 < t \le \bar{T}_4 \\ -a_{max} \sin^2(\omega\,(t - \bar{T}_4)) & \bar{T}_4 < t \le \bar{T}_5 \\ -a_{max} & \bar{T}_5 < t \le \bar{T}_6 \\ -a_{max} \sin^2(\omega\,(t - \bar{T}_x)) & \bar{T}_6 < t \le \bar{T}_x \\ 0 & \text{otherwise,} \end{cases} \quad (1$$

where

$$\bar{T}_1 = T_j/2$$
$$\bar{T}_2 = T_j/2 + T_a$$
$$\bar{T}_3 = T_j + T_a$$
$$\bar{T}_4 = T_j + T_a + T_v$$
$$\bar{T}_5 = 3/2\,T_j + T_a + T_v$$
$$\bar{T}_6 = 3/2\,T_j + 2T_a + T_v$$

and

$$\bar{T}_x = 2T_j + 2T_a + T_v, \quad (14)$$

with $\omega = \pi/T_j$ and $T_j/2$ denoting the time it takes until maximum acceleration a_{max} is reached. The time T_a is the period of constant maximum acceleration and T_v is the time of constant velocity. Then the whole scheme is applied a second time to the deceleration phase. Consequently, $\bar{T}_x$ is the total travel time for the distance Δx.

Integration of (13) yields the velocity $v_d(T_j, T_a, T_v, t) = \int_0^t a_d(T_j, T_a, T_v, \tilde{t})\,\mathrm{d}\tilde{t}$ and the position $x_d(T_j, T_a, T_v, t) = \int_0^t v_d(T_j, T_a, T_v, \tilde{t})\,\mathrm{d}\tilde{t}$, all of which are depicted in Figure 8. The time derivative of (13) yields the jerk, $j_d(t) = \dot{a}_d(t)$. It remains to compute the time periods T_j, T_a and T_v from the robot parameters maximum jerk j_{max}, acceleration a_{max} and velocity v_{max}.

The actuator needs the time $T_j/2$ to build up its maximum acceleration. This time span is consequently determined by the maximum jerk,

$$T_j = \pi\, a_{max}/j_{max}. \quad (15a)$$

Moreover, T_a computes from the time it takes to accelerate up to maximum velocity,

$$T_a = v_{max}/a_{max} - T_j/2, \quad (15b)$$

and T_v is determined by the distance Δx to be covered,

$$T_v = \frac{2\,\Delta x}{a_{max}\,(T_j + 2T_a)} - \frac{T_j^2 + 3T_jT_a + 2T_a^2}{T_j + 2T_a}. \quad (15c)$$

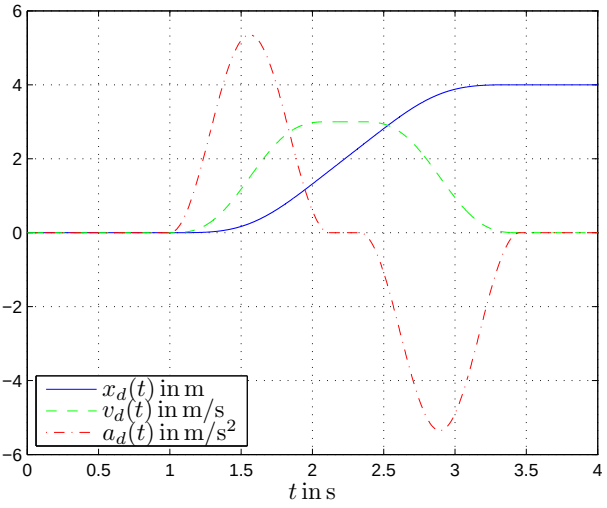

Fig. 8 Position $x_d(t)$, velocity $v_d(t)$ and acceleration profile $a_d(t)$ according to equation (13) for the (arbitrary) parameters $j_{max} = 15\mathrm{m/s^3}$, $a_{max} = 10\mathrm{m/s^2}$ and $v_{max} = 3\mathrm{m/s}$.

These equations only hold for a sufficiently large Δx, otherwise maximum velocity or even maximum acceleration is not reached. Therefore, the trajectory generator must distinguish these cases, which is easy to do and thus is omitted for brevity.

3.4 Processing Time per Panel Side

Each defect $\mathbf{D}_i$, $i = 1, \ldots, N$ is covered by a number of $n_{\mathbf{P}_i}$ patches. The processing time of one patch $\mathbf{p}_j$ consists of the positioning time $\bar{T}_j = \max(\bar{T}_{x_j}, \bar{T}_{y_j})$, with $\bar{T}_{x_j}$ and $\bar{T}_{y_j}$ according to (14), and the patching time T_p. Assuming that no disturbances are acting on the system, the total processing time per panel side is the sum of all patch processing times

$$\bar{\bar{T}} = \sum_{i=1}^{N} \left(n_{\mathbf{P}_i}\, T_p + \sum_{j=1}^{n_{\mathbf{P}_i}} \bar{T}_j \right). \quad (16)$$

The Scheduling Agent associates each panel side with its estimated processing time $\bar{\bar{T}}$ and uses this information for workload balancing.

4 Simulation Results

This section briefly demonstrates the efficiency of the proposed algorithms and points out their pros and cons.

4.1 Patch Placement

Figure 9 exemplarily shows the potential of the patch placement algorithm. The optimal solution requires six

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

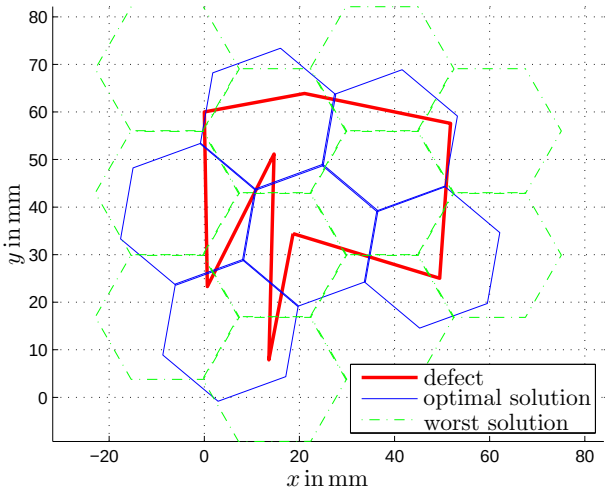

Fig. 9 The patch placement algorithm reduces the number of required patches for the defect to a global minimum of six.

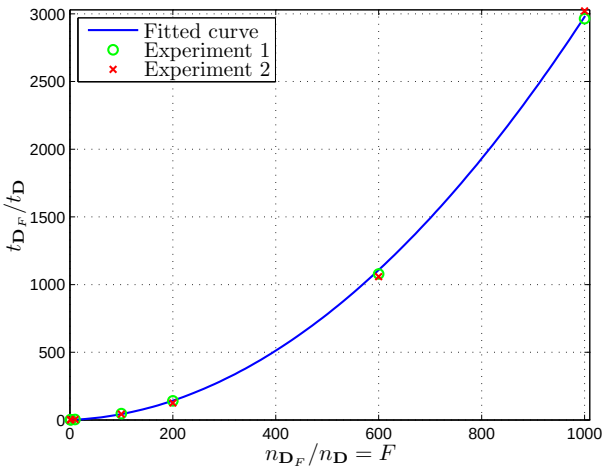

Fig. 10 Normalized computation time $t_{\mathbf{D}_F}/t_{\mathbf{D}}$ versus normalized problem size $F = n_{\mathbf{D}_F}/n_{\mathbf{D}}$.

patches to cover the defect as compared to eleven patches of the worst solution. Major advantages of this approach are: First, the above algorithm does not impose any restrictions on the shape of the defect polygon, in particular convexity is not required. Second, a global minimum with respect to the positioning accuracy is found. Third, any number of arbitrary constraints can be added. If one of these constraints is not met, the solution is not feasible and thus ignored. This is very important, in particular with respect to the feasibility of the patch arrangement. Fourth, the algorithm can be terminated any time, which is essential for hard real-time constraints. Then the best solution found so far is going to be worked with. The major drawback of this algorithm is its rather high computational cost.

The time complexity of an algorithm depends on the problem size n, which, according to Sipser (2012), is equal to the length of the input string. In case of the patch placement algorithm presented in Section 3.1 n is equal to the number of nodes $n_{\mathbf{D}}$ of the defect polygon $\mathbf{D}$. In order to analyze the time complexity of the patch placement algorithm the following experiment is carried out.

To begin with, polygons of the same shape and size, but with a different number of nodes are created. This is achieved by adding nodes exactly at the edges of the original polygon. Suppose an edge of the original polygon is given by the two nodes $\mathbf{d}_j$ and $\mathbf{d}_{j+1}$. Then the number $n_{\mathbf{D}}$ of nodes of the original polygon is increased by a factor F by inserting $(F-1)$ nodes

$$\mathbf{d}_{j,\,f} = \mathbf{d}_j + f\,(\mathbf{d}_{j+1} - \mathbf{d}_j)/F,\ f = \{1, \ldots, F-1\} \quad (17)$$

into every edge of the polygon. This way, the polygons $\mathbf{D}_2$, $\mathbf{D}_{10}$, $\mathbf{D}_{100}$, $\mathbf{D}_{200}$, $\mathbf{D}_{600}$ and $\mathbf{D}_{1000}$ are created, for the factors $F \in \{2, 10, 100, 200, 600, 1000\}$.

Then, the patch placement algorithm is applied to these polygons. It goes through exactly the same computation steps for each of the defect polygons since they are of identical shape and size. Only the input size, i.e. the number of nodes $n_{\mathbf{D}}$, varies. The result is depicted in Figure 10. It shows the computation time of the patch placement algorithm $t_{\mathbf{D}_F}$ normalized to the computation time $t_{\mathbf{D}}$ of the original defect polygon over the normalized problem size $F = n_{\mathbf{D}_F}/n_{\mathbf{D}}$. Two examples are chosen, Experiment 1 refers to a rather small defect requiring only 3 patches, whereas Experiment 2 is carried out with the big defect shown in Figure 9. The absolute computation time $t_{\mathbf{D}}$ of the original defects is 0.07s and 0.22s, respectively.

Clearly, the experiments reveal a quadratic dependence of the computation time on the problem size, in particular a least-squares curve fit of the experiments yields

$$\frac{t_{\mathbf{D}_F}}{t_{\mathbf{D}}} = c_2 \left(\frac{n_{\mathbf{D}_F}}{n_{\mathbf{D}}}\right)^2 + c_1 \frac{n_{\mathbf{D}_F}}{n_{\mathbf{D}}} + c_0 \quad (18)$$

with constants $c_2 = 0.0025$, $c_1 = 0.1479$ and $c_0 = 0.0028$.

Among all optimization steps, the minimization of the number of patches has the largest contribution to savings of both production time and wood. Each patch requires approximately 4s of production time, including positioning and the patching action itself. Also the maximum number of patches allowed per defect is limited due to quality reasons, which is why minimizing the number of patches needed per defect enables to process even lower quality raw material and as a consequence

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

reduces the amount of rejects. If only one defect of a panel is too big to patch, the entire panel, i.e. up to 20kg of wood, needs to be rejected. In conclusion, it is worth computing the global minimum of patches required for each defect, even at higher computational costs.

Similar problems to patch placement also arise in network covering of telecommunications. In Das et al. (2006), a given number of base stations of variable, but identical transmission range shall be distributed in a convex region such that the required total transmission power is minimized. This boils down to a covering problem, where the number of circles of equal radius is given a priori, but their location needs to be chosen optimally as to minimize the radius. Compared to the patch placement problem, the optimization variables are the same, i.e. the position of the circles. However, the objective function and the parameter of the algorithm, i.e. the number of circles and the circle radius, are interchanged.

4.2 Path Planning

In the following example, the cost of each arc corresponds to the Euclidean distance between the respective two nodes. Figure 11 depicts a 1.5m long panel exhibiting 24 "small" and 3 "big" defects (at approximately $x = 0.35$m, 0.8m, 0.9m). The small defects are covered by one patch, whereas the big ones require three patches each. For the path planning algorithm, the respective patches are clustered and considered as one node. The clustering is indicated by the black lines connecting the nodes.

The solution of the simplest possible path planning method "from-left-to-right" (L2R) is compared to the results of the Ant Colony Algorithm (ACOA) and the Local Search Receding Horizon Algorithm (LSRHA) as presented in Sections 3.2.1 and 3.2.2 respectively. Although the L2R path already yields quite good results, one can easily identify its weakness. If nodes are very close in x-direction but far from each other in y-direction, this method generates solutions that permanently jump up and down along the y-axis. Thus, this method becomes less effective as the ratio between panel length and width decreases.

Provided a computation time of 5ms, the ACOA manages to improve the start solution from $J_{L2R} = 4.6$m to $J_{ACO} = 3.7$m. However, the LSRHA finds a path of length $J_{LSRH} = 3.5$m in only 1ms. The reason for the high performance of the LSRHA is that it makes perfect use of the peculiarities of the traveling salesman problem at hand. First, there is a dedicated production flow, and second, the panels are much longer than wide.

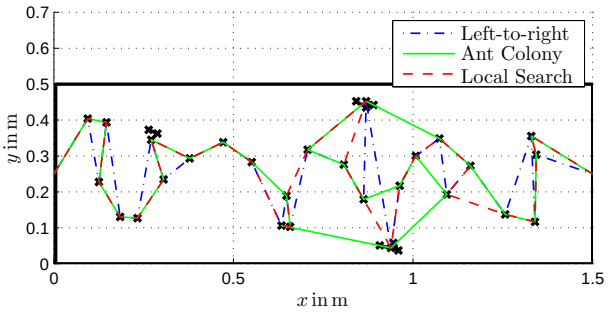

Fig. 11 Results of three path planning strategies for an exemplary panel.

So it is easy to create good start solutions which only require local refinement[9]. The LSRHA is perfectly suited to correct these errors.

Therefore, current benchmark algorithms for classical traveling salesman problems (or other complex combinatorial optimization problems) are implemented as a two-stage-process. As a first step, one makes use of the ACOA's excellent global search ability to generate good start solutions which, in a second step, are refined using local search, see Dorigo and Stützle (2004).

In the considered application, hard real-time constraints have to be met. Therefore, the possibility to simply stop the algorithm after a given time is a vital backup system. Both, the ACOA and the LSRHA can be terminated any time, since they find good paths very quickly and the best-so-far path is remembered. The drawback of these heuristic methods is that optimality of the computed path cannot be proven. Nevertheless, the above mentioned advantages make the LSRHA the method of choice for this particular path planning problem.

Regarding general applicability, it should be noted that these algorithms can be applied to any robot design as nothing but the cost matrix would change. Going beyond the traveling salesman problem, both of these algorithms are applicable to any problem that can be stated in the form as described in Dorigo and Di Caro (1999). Various examples for the application of ACOAs and Local Search Algorithms are given in Gambardella and Dorigo (2000); Chu et al. (2004); Gutjahr and Rauner (2007). The Receding Horizon Concept is restricted to problems exhibiting a distinguished direction of propagation, e.g. in the form of a dedicated process direction as in the application at hand or in a timely fashion as in air traffic management.

[9] It is worth mentioning that the most suitable start solution for the LSRHA is the nearest-neighbor path. Usually, the nearest-neighbor path is extremely good except for a few nodes that are left behind.

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

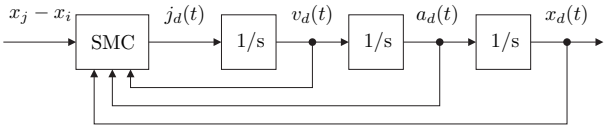

Fig. 12 Schematics of the time-optimal Bang-Bang-Trajectory-Generator.

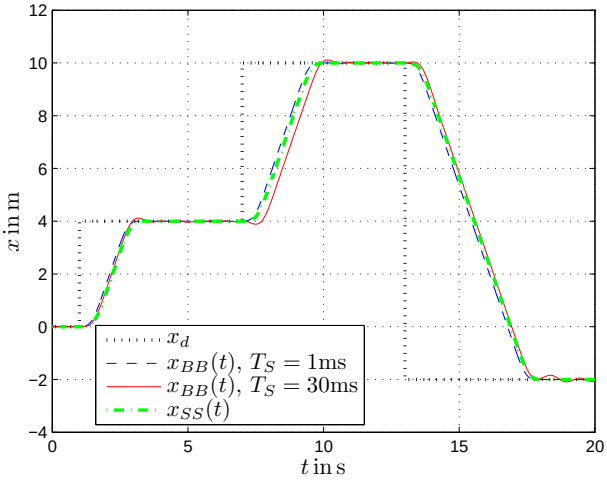

Fig. 13 Comparison of the SSTG to the BBTG for different sampling times. The parameters remain the same as in Figure 8.

To the best of our knowledge, this is the first application of ACOA and LSRHA to process sequencing optimization in wood patching applications.

4.3 Trajectory Generation

The problem of finding a suitable trajectory is a well established problem, see, e.g., Biagiotti and Melchiorri (2008). In this book an online time-optimal trajectory generator is outlined. Basically, it consists of a chain of three integrators and a time-optimal sliding mode controller (SMC) driving them towards their respective reference values. Since this controller permanently switches between maximum and minimum jerk, it shall be referred to as Bang-Bang-Trajectory-Generator (BBTG), see Figure 12. For more details refer to Zanasi and Morselli (2002). The BBTG is easy to implement and therefore serves as a basis for comparison to the Sine-Square-Trajectory-Generator (SSTG), presented in Section 3.3, see Figure 13.

In the (quasi-) time-continuous case, sampling time $T_S = 1$ms, the BBTG generates strictly time-optimal trajectories. However, it should be noted that the BBTG is only able to switch the desired jerk at the sampling times. For higher sampling times, this results in oscillating behavior, which clearly counteracts precise and fast positioning. Consequently, time-optimality is lost. The more recent papers Gerelli and Guarino Lo Bianco (2010); Guarino Lo Bianco and Ghilardelli (2012) present solutions to this problem.

The SSTG yields trajectories $x_{SS}(t)$ that are nearly time-optimal, yet it has some advantages over the BBTG. First, the SSTG yields smooth trajectories without overshoot or oscillation, even for the time-discrete case. Second, the SSTG is suited for offline processing time estimation, since the estimation of the travel time is reduced to computing the time periods T_j, T_a and T_v according to (15). On the contrary, the BBTG is designed for online trajectory generation, meaning each simulation time-step has to be computed in order to estimate the overall trajectory time. For the application at hand, the above mentioned advantages of SSTG are decisive. In particular, the smoothness of the SSTG-trajectories is a great benefit for the application at hand since the panels are moved merely via friction forces.

Finally, it shall be noted that both trajectory generators can be applied to any 1D-motion-planning-task. This naturally includes multidimensional motion that can be decomposed accordingly.

4.4 Processing Time Estimation

In order to provide significant simulation results for the processing time estimation a random test set of 500 panels, i.e. 1000 panel sides, is generated. This test set incorporates all statistical data available[10] for the panels, e.g. panel length distribution, defect distribution per square meter, percentage of defects requiring $n_{\mathbf{P}} = \{1, 2, 3, \ldots\}$ patches, etc.

Using the afore presented algorithms, the processing time of each panel side is estimated. The result is shown in Figure 14. Since the panel processing time is subject to large variations, scheduling algorithms are needed to evenly distribute the work load between the three patching lines sketched in Figure 3. As already mentioned in the introduction, this task can be formulated as classical *parallel machine scheduling with a single server*.

5 Conclusions

In this paper, a novel layout of a fully automated patching plant for shuttering panels is presented. In comparison to existing solutions, legacy equipment of the current semi-automatic patching process is reused in this process.

[10] The data is provided by our partner saw mill Lip Bled, see http://en.lip-bled.si

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

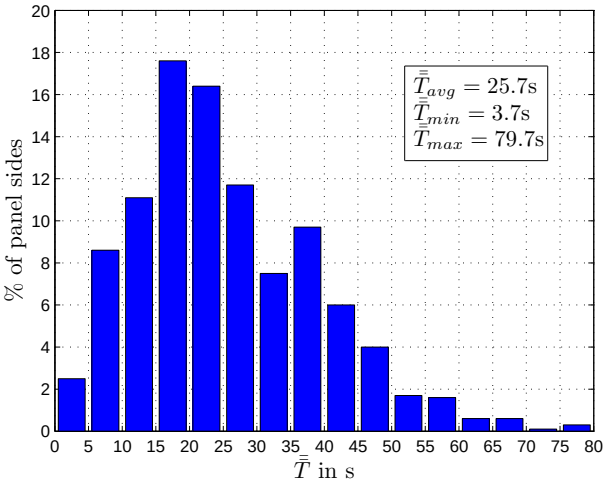

Fig. 14 Distribution of the processing time $\bar{\bar{T}}$ per panel side grouped in intervalls of 5s.

The core of this paper deals with optimizing the individual production steps of panel processing, namely patch placement, path planning and trajectory generation. The patch placement algorithm is based on the idea of hexagonally closest packaging. The defect polygon is put in the middle of a hexagon tesselation and displaced relatively to it until the defect is covered by a minimum number of patches. Thus, within the closed solution space, a global minimum with respect to the positioning accuracy of the patching robot is found. It is computationally rather expensive, but has great potential for not only saving production time but also wood, thus increasing the yield.

For the path planning problem, a Local Search Algorithm is employed. It randomly exchanges λ arcs of a given start solution until a minimum is found. In the sense of λ-optimality, increasing λ yields increasingly strong necessary conditions for optimality. In combination with a Receding Horizon Concept which divides the entire problem instance into several small subproblems, this strategy yields superior results compared to the frequently used Ant Colony Algorithm. However, this is mainly due to the nature of the problem, namely the dedicated production flow, the long but slim panels and the rather small problem size.

The trajectory generation problem is tackled with the so-called Sine-Square-Trajectory-Generator, which prescribes the acceleration in form of a smooth curve, thus aiding in precise positioning. Moreover, the computational costs for this approach are very low, since it comes down to solving algebraic equations. This is also why it is particularly suited for offline-computation. Compared to the Bang-Bang-Trajectory-Generator it is not strictly time-optimal, but in practice the difference in trajectory times is negligible.

Based on the results of the above algorithms, the production time of the individual panels can be estimated. The wide range of processing times observed for the individual panels clearly shows the potential of sophisticated scheduling strategies, which will be a part of future work in the Hol-I-Wood PR project.

For the implementation of the algorithms in the overall automation system, a Multi-Agent Approach is employed. The foundations of this Multi-Agent Architecture have already been implemented on a hardware system located in the laboratory facilities of the Automation and Control Institute (ACIN) at Vienna University of Technology, see Vallée et al. (2009). Moreover, the pilot patching robot is currently being set up and the overall automation concept is going to be implemented on a pilot plant in the near future.

Acknowledgements The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 284573.

References

F. Allgöwer, T.A. Badgwell, J.S. Qin, J.B. Rawlings, S.J. Wright, Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview, in *Advances in Control: Highlights of ECC '99*, ed. by P.M. Frank (Springer, London, UK, 1999), pp. 391–449

D.L. Applegate, R.E. Bixby, V. Chvátal, W.J. Cook, *The Traveling Salesman Problem: A Computational Study.* Princeton Series in Applied Mathematics (Princeton University Press, Princeton, NJ, USA, 2006)

M. Barbati, G. Bruno, A. Genovese, Applications of agent-based models for optimization problems: A literature review. Expert Systems with Applications **39**(5), 6020–6028 (2012)

L. Biagiotti, C. Melchiorri, *Trajectory Planning for Automatic Machines and Robots* (Springer, Berlin, Heidelberg, GER, 2008)

S. Bussmann, N.R. Jennings, M. Wooldridge, *Multiagent Systems for Manufacturing Control: A Design Methodology.* Springer Series on Agent Technology (Springer, Berlin, Heidelberg, GER, 2004)

S.-C. Chu, J.F. Roddick, C.-J. Su, J.-S. Pan, Constrained Ant Colony Optimization for Data Clustering, in *Proc. 8th Pacific Rim International Conference on Artificial Intelligence*, Auckland, NZ, 2004, pp. 534–543

G.K. Das, S. Das, S.C. Nandy, B.P. Sinha, Efficient algorithm for placing a given number of base stations to cover a convex region. Journal of Parallel and Distributed Computing **66**(11), 1353–1358 (2006)

M. Dorigo, G. Di Caro, Ant Colony Optimization: A New Meta-Heuristic, in *Proc. Congress on Evolutionary Computation*, vol. 2, Washington D.C., USA, 1999, pp. 1470–1477

M. Dorigo, T. Stützle, *Ant Colony Optimization* (MIT Press, Cambridge, MA, USA, 2004)

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

A.E. Eiben, J.E. Smith, *Introduction to evolutionary computing*. Natural Computing Series (Springer, Berlin, Heidelberg, GER, 2003)

L.M. Gambardella, M. Dorigo, An ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS Journal on Computing **12**(3), 237–255 (2000)

M. Gen, L. Lin, Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. Journal of Intelligent Manufacturing, 1–18 (2013). doi:10.1007/s10845-013-0804-4

O. Gerelli, C. Guarino Lo Bianco, A Discrete-Time Filter for the On-Line Generation of Trajectories with Bounded Velocity, Acceleration, and Jerk, in *Proc. IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, 2010, pp. 3989–3994

C. Guarino Lo Bianco, F. Ghilardelli, Third Order System for the Generation Of Minimum-Time Trajectories with Asymmetric Bounds on Velocity, Acceleration, and Jerk, in *Proc. International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, 2012, pp. 137–143

W.J. Gutjahr, M.S. Rauner, An aco algorithm for a dynamic regional nurse-scheduling problem in austria. Computers & Operations Research, Special Issue: Logistics of Health Care Management **34**(3), 642–666 (2007)

N.G. Hall, C.N. Potts, C. Sriskandarajah, Parallel machine scheduling with a common server. Discrete Applied Mathematics **102**(3), 223–243 (2000)

I. Hegny, O. Hummer, A. Zoitl, G. Koppensteiner, M. Merdan, Integrating software agents and IEC 61499 realtime control for reconfigurable distributed manufacturing systems, in *Proc. International Symposium on Industrial Embedded Systems*, La Grande-Motte, FR, 2008, pp. 249–252

K. Helsgaun, An effective implementation of the lin–kernighan traveling salesman heuristic. European Journal of Operational Research **126**(1), 106–130 (2000)

K. Helsgaun, General k-opt submoves for the linkernighan tsp heuristic. Mathematical Programming Computation **1**(2-3), 119–163 (2009)

X.-B. Hu, W.-H. Chen, Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. Engineering Applications of Artificial Intelligence **18**(5), 633–642 (2005a)

X.-B. Hu, W.-H. Chen, Receding horizon control for aircraft arrival sequencing and scheduling. IEEE Transactions on Intelligent Transportation Systems **6**(2), 189–197 (2005b)

X.-B. Hu, W.-H. Chen, E. Di Paolo, Multiairport capacity management: Genetic algorithm with receding horizon. IEEE Transactions on Intelligent Transportation Systems **8**(2), 254–263 (2007)

M.N. Huhns, M.P. Singh, M. Burstein, K. Decker, K.E. Durfee, T. Finin, T.L. Gasser, H. Goradia, P.N. Jennings, K. Lakkaraju, H. Nakashima, H. Van Dyke Parunak, J.S. Rosenschein, A. Ruvinsky, G. Sukthankar, S. Swarup, K. Sycara, M. Tambe, T. Wagner, L. Zavafa, Research directions for service-oriented multiagent systems. IEEE Internet Computing **9**(6), 65–70 (2005)

Jade, *Java Agent DEvelopment Framework*, Available from: http://jade.tilab.com/, 2014. [23 July 2014]

N.R. Jennings, K. Sycara, *A Roadmap of Agent Research and Development*, 1998

M.-Y. Kim, Y.H. Lee, Mip models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. Computers & Operations Research **39**(11), 2457–2468 (2012)

K. Kouiss, H. Pierreval, N. Mebarki, Using multi-agent architecture in fms for dynamic scheduling. Journal of Intelligent Manufacturing **8**(1), 41–47 (1997)

S.A. Kravchenko, F. Werner, Parallel machine scheduling problems with a single server. Mathematical and Computer Modelling **26**(12), 1–11 (1997)

P. Leitão, Agent-based distributed manufacturing control: A state-of-the-art survey. Engineering Applications of Artificial Intelligence **22**(7), 979–991 (2009)

W. Lepuschitz, A. Zoitl, M. Vallee, M. Merdan, Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **41**(1), 52–69 (2011)

S. Lin, Computer solutions of the traveling salesman problem. Bell System Technical Journal **44**(6), 2245–2269 (1965)

M. Melik-Merkumians, T. Baier, M. Steinegger, W. Lepuschitz, I. Hegny, A. Zoitl, Towards OPC UA as portable SOA Middleware between Control Software and External Added Value Applications, in *Proc. IEEE 17th International Conference on Emerging Technologies and Factory Automation*, Krakow, PL, 2012, pp. 1–8

M. Merdan, T. Moser, D. Wahyudin, S. Biffl, P. Vrba, Simulation of Workflow Scheduling Strategies Using the MAST Test Management System, in *Proc. 10th International Conference on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, 2008, pp. 1172–1177

M. Mucientes, J.C. Vidal, A. Bugarin, M. Lama, Processing times estimation in a manufacturing industry through genetic programming, in *Proc. 3rd International Workshop on Genetic and Evolving Systems*, Witten-Bommerholz, GER, 2008, pp. 95–100

L. Ollinger, D. Zuhlke, A. Theorin, C. Johnsson, A reference architecture for service-oriented control procedures and its implementation with SysML and Grafchart, in *Proc. IEEE 18th Conference on Emerging Technologies Factory Automation*, Cagliari, IT, 2013, pp. 1–8

I. Or, E. Duman, Optimization issues in automated production of printed circuit boards: operations sequencing, feeder configuration and load balancing problems, in *Proc. IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1, Kauai, Hawaii, 1996, pp. 227–232

M. Pěchouček, V. Mařík, Industrial deployment of multi-agent technologies: review and selected case studies. Autonomous Agents and Multi-Agent Systems **17**(3), 397–431 (2008)

L. Ribeiro, J. Barata, P. Mendes, MAS and SOA: Complementary Automation Paradigms, in *Innovation in Manufacturing Networks* (Springer, NY, USA, 2008), pp. 259–268

J. Rubinovitz, R.A. Wysk, Task level off-line programming system for robotic arc welding - an overview. Journal of Manufacturing Systems **7**(4), 293–306 (1988)

W. Shen, L. Wang, Q. Hao, Agent-based distributed manufacturing process planning and scheduling: A state-of-the-art survey. IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews **36**(4), 563–577 (2006)

M. Sipser, *Introduction to the Theory of Computation* (Cengage Learning, Boston, MA, USA, 2012)

S.O. Tasan, S. Tunali, A review of the current applications of genetic algorithms in assembly line balancing. Journal of Intelligent Manufacturing **19**(1), 49–69 (2008)

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z

M. Vallée, H. Kaindl, M. Merdan, W. Lepuschitz, E. Arnautovic, P. Vrba, An Automation Agent Architecture with A Reflective World Model in Manufacturing Systems, in *Proc. IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, USA, 2009, pp. 305–310

M. Vincze, G. Biegelbauer, A. Pichler, Painting parts automatically at lot size one, in *Proc. International Workshop on Robot Sensing*, Graz, AUT, 2004, pp. 35–40

P. Vrba, V. Mařík, Capabilities of dynamic reconfiguration of multiagent-based industrial control systems. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans **40**(2), 213–223 (2010)

R. Williams, *The Geometrical Foundation of Natural Structure: A Source Book of Design* (Dover Publications, New York, USA, 1979)

F. Zambonelli, N.R. Jennings, A. Omicini, M. Wooldridge, Agent-Oriented Software Engineering for Internet Applications, in *Book Coordination of Internet Agents: Models, Technologies and Applications* (Springer, Heidelberg, GER, 2000), pp. 326–346

R. Zanasi, R. Morselli, Third Order Trajectory Generator Satisfying Velocity, Acceleration and Jerk Constraints, in *Proc. International Conference on Control Applications*, vol. 2, Glasgow, Scotland, UK, 2002, pp. 1165–1170

Z.-H. Zhan, J. Zhang, Y. Li, O. Liu, S.K. Kwok, W.H. Ip, O. Kaynak, An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem. IEEE Transactions on Intelligent Transportation Systems **11**(2), 399–412 (2010)

Post-print version of the article: M. W. Hofmair, M. Melik-Merkumians, M. Böck, M. Merdan, G. Schitter, and A. Kugi, "Patching process optimization in an agent-controlled timber mill", *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 69–84, 2017, ISSN: 1572-8145. DOI: 10.1007/s10845-014-0962-z