Int J Adv Manuf Technol (2013) 69:2555–2568
DOI 10.1007/s00170-013-5185-5

ORIGINAL ARTICLE

Task planning for cooperating self-reconfigurable mobile fixtures

Włodzimierz Kasprzak · Dimiter Zlatanov · Wojciech Szynkiewicz · Teresa Zielińska

Received: 23 February 2013 / Accepted: 5 July 2013 / Published online: 13 August 2013

Abstract The work solves the problem of task and motion planning of a self-reconfigurable fixture system. A feasible solution is a key requirement for the viability of such systems, which have raised hopes of overcoming the deficiencies that more traditional fixtures are recognized to have in the dynamic conditions of modern manufacturing, with its increasing emphasis on flexibility, adaptability, and automation. The paper proposes an application-independent approach for the generation of a time-relevant action plan for the locomotion, reconfiguration, and positioning of two or more mobile robotic fixtures. The fixture agents need to provide local support for a large workpiece during machining. The path-planning problem is converted into a constraint satisfaction problem (CSP). The proposed approach is called Triple-CSP, as it applies an incremental state search to solve three hierarchical path-planning tasks for the three components of each mobile fixture agent: a supporting head, a mobile base, and a parallel manipulator. A final time-related trajectory (time scaling of actions) for the agents' entire task execution is obtained. Thus, the planner takes into account all the relevant physical, geometrical, and time-related constraints.

Keywords Active supports · Agent system · Constraint satisfaction · Hole drilling · Milling · Mobile fixtures · Robotized machining · Task planning

W. Kasprzak (✉) · W. Szynkiewicz
Institute of Control and Computation Engineering,
Warsaw University of Technology, ul.Nowowiejska 15/19,
00-665 Warsaw, Poland
e-mail: W.Kasprzak@elka.pw.edu.pl

W. Szynkiewicz
e-mail: W.Szynkiewicz@elka.pw.edu.pl

D. Zlatanov
PMAR Laboratory, DIMEC, University of Genoa,
Via all'Opera Pia 15a, 16145 Genoa, Italy
e-mail: zlatanov@dimec.unige.it

T. Zielińska
Institute of Aeronautics and Applied Mechanics,
Warsaw University of Technology, ul.Nowowiejska 24,
00-665 Warsaw, Poland
e-mail: teresaz@meil.pw.edu.pl

1 Introduction

A fixture is a device for locating, constraining, and adequately supporting a deformable workpiece during a manufacturing operation [1–3]. Components made of thin metallic or composite sheets are ubiquitous in automotive and aircraft manufacturing. For both esthetic and economic reasons, they are becoming increasingly common in other sectors. The fixtures traditionally used in the manufacturing of thin-sheet metal parts are large moulds reproducing the shape of the skin to be supported, but this type of fixture is part-specific and not reconfigurable. Modular flexible fixture systems (MFFSs) and single-structure flexible fixture systems (SSFFSs) are intended to make retooling easer by providing limited reconfigurability.However, most FFSs still require some human intervention to reconfigure and few are suitable for thin sheets [4–6]. One attempt to automate reconfiguration is the use of an SSFFS of the pin-bed type, with a matrix of supports, which provides support comparable to a mould-like fixture. The main disadvantages are the high cost, as well as a reduced ability to adapt easily and efficiently to parts of different sizes, drawbacks in part linked to a lack of modularity.

Robotic fixtureless assemblies (RFAs) replace traditional fixtures by robot manipulators equipped with grippers that

Fig. 1 Two cooperating mobile agents supporting a workpiece in the SwarmItFIX project

can cooperatively hold the workpiece [7, 8]. Using RAFs, different parts can be manufactured within one work cell and transitions to other workpieces can be done relatively quickly. However, existing and proposed RAFs are suited only for rigid and relatively small parts, as they use a limited number of traditional nonmobile robots.

In keeping with current trends in manufacturing [9], fixture systems are gradually being transformed to allow increased flexibility, reconfigurability, and automation [10]. Responding to such needs, a self-reconfigurable fixture system, primarily targeted at the aerospace industry, has been developed within the inter-European SwarmItFIX project [11, 12]. The fixturing strategy uses mobile parallel manipulators (so-called parallel kinematic machines or PKMs) continuously repositioning to provide support to the thin-sheet workpiece near the moving machine tool (Fig. 1).

The SwarmItFIX fixturing system combines the advantages of RFAs with those of MFFSs, namely, ability to distribute the support action, adaptability to part shapes in a larger range, and high stiffness of the provided support. Each fixture element, referred to as a physical agent, is composed of three distinct parts: a mobile robot base, a PKM fixed to the mobile platform, and a supporting head with shape adaptation and adhesion capabilities (a novel design using phase-change magneto-rheological fluid and vacuum suction). The mobility of each fixturing agent and the resulting possibility for the agents to group in regions where a machining process is being executed, as well as the ability of the supporting head to adapt to the part's local geometry, ensure good fixturing support with fewer agents for a large variety of workpieces and manufacturing scenarios. This novel solution is seen as more cost-effective when compared to both the use of traditional solid moulds and existing reconfigurable fixtures with numerous extendable rods.

A key issue for the viability of such an autonomous fixturing system is the ability to reliably and automatically generate complete time-dependent task and motion plans for the agents' actions [13]. In view of the clear need and potential of mobile-agent-based self-reconfigurable systems, solving the arising complex planning problem becomes a research goal of significant importance. The present paper provides the required solution.

As planning involves the generation of a sequence of fixturing configurations, the problem is to some degree related to the traditional fixture design. Although research devoted to fixture design and optimization is extensive [1, 2, 14, 15], it invariably assumes passive fixtures and static planning conditions. Recently, the problem of active fixture design is attracting increased attention, e.g., studying changing conditions due to moving loads [16].

Various techniques have been proposed for optimization of fixture layout by formulating different objective functions to determine the location of fixturing supports. In the research for compliant sheet metal parts, Menassa and De Vries [1] use a finite element model of the workpiece to model the deformation, and determine fixture locations by optimizing an objective that is a function of the deformations at the nodes. The design variables are three fixture locators on primary datum as required by the "3-2-1" principle. In [17], an optimization algorithm to obtain the optimal number and location of clamps that minimize the deformation of compliant parts is proposed. Cai et al. [2] propose the "N-2-1" fixture layout principle for constraining compliant sheet metal parts. This is used instead of the conventional "3-2-1" principle to reduce deformation of sheet metal parts. They present algorithms for finding the best N locating points such that total deformation of a sheet metal is minimized. They use a finite element model of the part with quadratic interpolation, constraining nodes in contact with the primary datum to only in-plane motion. Nonlinear programming is utilized to obtain the optimal fixture layout. DeMeter [14] introduces a fast support layout optimization model to minimize the maximum displacement-to-tolerance ratio of a set of part features subject to a system of machining loads. The speedup of the optimization is obtained by a reduced stiffness matrix approach.

Similarly, extensive literature exists on robot motion planning. Thus, combinatorial motion-planning algorithms construct a discrete representation that exactly represents

the original C-space [18–20]. This leads to complete planning approaches, which are guaranteed to find a solution when it exists, or correctly report failure if one does not exist.

However, this may result in an excessive computational burden in high dimensions, and in environments described by a large number of obstacles. Avoiding such a representation is the main underlying idea leading to the development of sampling-based algorithms. The most effective robot motion-planning strategies today are built upon sampling-based techniques, including the Probabilistic Roadmap Methods (PRMs) [21] and Rapidly-exploring Randomized Trees (RRTs) [22] and their variants. The general methodology of sampling-based algorithms is to construct a graph (the roadmap) during preprocessing to capture the connectivity of the valid subset of the C-space and then to query the road map to find a path for a given motion-planning task [20, 23]. However, these algorithms may not terminate when no collision-free path exists in the free space and may sometimes fail to find a path, especially when one exists. Although the guiding ideas in the above path-planning approaches can inform the search of the solution of the stated problem, none can be applied directly. One reasons is the higher complexity: note for example that in our case, a new robot-planning problem must be solved at every repositioning step and the target pose of the end effector is also to be determined. On the other hand, the rigorous and numerous manufacturing requirements go beyond the usual obstacle avoidance constraints. Robot motion planning can be also viewed as an optimization problem that aims to minimize a given objective function. Numerous approaches have been proposed to solve this problem [20, 24–26].

In the case of SwarmItFIX, the planning task concerns much more than the classical motion-planning problem for a single robot. What is required is a timed sequence of supporting-head placements, at acceptable distances, neither too far nor too near, to both the milled contour and each other. To realize these placements, suitable and collision-free trajectories must be found, for both the robot bases discretely moving on the bench and for the parallel manipulators' changes of configuration. Thus, the problem that would be named path and motion planning in a conventional robotic system, for SwarmItFIX, is decomposed into three hierarchical subproblems: head placements, mobile base positioning, and manipulator motion.

For a highly reconfigurable system such as SwarmItFIX, the standard global optimization approach to both robot and fixture planning becomes of questionable validity because of the presence of numerous, variable, time-dependent, and noncommensurate criteria. While it may be possible to reasonably select a single scalar objective function when the fixturing problem is narrowly defined in terms of the type of the part, the fixture, and the process, as well as with respect to time and space, such a choice becomes arbitrary in a dynamic industrial scenario where competing and highly variable desiderata arise.

On the other hand, in high-precision industrial applications, manufacturers have rigorous and detailed specifications which (a) have to be satisfied without compromise or mutual compensation and (b) leave only limited possibility for further optimization and improvement of the process quality through planning. In brief, what manufacturers seek is a task plan that will unfailingly satisfy all requirements exactly as specified, not one that will endeavor to "improve" on them by prioritizing a single parameter above the others. Thus, the emphasis shifts from the search of an optimum to the satisfaction of constraints. Therefore, a novel but natural approach to such a planning task is to view it as a constraint satisfaction problem (CSP), a well-known method in "Artificial Intelligence" [27]. We consider the path-planning task as a finite domain CSP: given a set of variables, together with a finite set of possible values that can be assigned to each variable, and a list of constraints, find particular values of all the variables (i.e., a complete assignment) that satisfy every constraint. A variety of approaches can be used to solve a CSP. Integer-programming techniques (cutting plane methods and branch and bound) can be applied to find an exact solution [28]. On the other hand, there are various approaches that provide an approximate solution, including local search, simulated annealing, genetic algorithms, etc. These approaches usually try to improve an initial complete assignment to CSP variables. A CSP algorithm can be designed to find just one feasible solution, with no preference as to which one, all solutions, or an optimal (or at least a good) solution, given some objective function defined in terms of some or all of the variables. Although the aim of typical algorithms for solving CSPs is to find a feasible solution, they can be adapted to finding an optimal solution. For example, a new (objective) constraint is introduced specifying that the value of the objective variable must be better than in the initial or previous solution.

By constraint programming, we mean the computer implementation of an algorithm for solving CSPs [29]. Here, the focus is on general purpose form and efficiency of the implementation.

There is a specific technique that is widely used for solving CSPs—it uses tree search combined with backtracking, forward checking (constraint propagation), and maintaining arc consistency [27]. Each node of the search tree corresponds to a partial solution in which the values of some variables are determined but the values of other variables remain to be decided. A transition to the next node means an assignment to a yet nondetermined

variable. We consider such a search algorithm with incremental assignment strategy to be ideally suitable to solve the path-planning task of active supports in SwarmItFIX. In general, planning is the process of finding a sequence of actions that transfer the world from some initial state to a desired state [30]. A general purpose (reusable) solution to this problem can be efficiently achieved by using constraint satisfaction search techniques. In this paper, we define a path-planning problem for each of the three agent parts, the supporting head, the mobile base, and the PKM posture, in terms of a CSP. Thus, the same computing tool, an incremental CSP search algorithm, can be used to solve the three parts of the planning problem. The three CSP solution procedures are referred to as Head-CSP, Base-CSP, and PKM-CSP. For greater efficiency, the three CSP searches are organized hierarchically. This structure allows to verify single assignments within Head-CSP, by assignments within Base-CSP, which are then in turn verified by assignments within the PKM-CSP. To the best of our knowledge, the use of such hierarchical CSP searches is a novel technique.

Let us also observe that in our approach, the manipulator joints paths (e.g., [31]) will be solved directly by the control unit of a manipulator [32] while the planner provides and checks goal positions only.

The remainder of the paper is organized as follows: The planning problem for cooperating fixturing agents is formulated in Section 2. The CSP-based planner is described in Section 3. The final time trajectory search is presented in Section 4. Examples of task plans in milling or drilling processes are given in Section 5. Final conclusions are drawn in Section 6.

2 Problem definition

A typical application scenario of the SwarmItFIX project and, in particular, the role of our planner is illustrated in Fig. 2. A stationary thin-sheet workpiece is subjected to a sequence of processes (such as milling, holing, or drilling) by a moving machine tool. The part is held in place by a limited number of static fixtures (not shown) that can support its weight but cannot adequately resist the large machining forces. The latter task is fulfilled by two (or more) mobile robotic agents that follow the progress of the machine tool and alternate to provide support of the piece in the immediate vicinity of the machined area. The purpose of the planner is to generate a path and time plan for every part of every mobile agent that satisfies given workpiece-machining requirements, that is collision-free, and that guarantees that the agents are timely following the computer numerical control (CNC) tool.

2.1 Cooperating mobile fixturing agents

Every mobile unit, referred to as a physical agent, is composed of a mobile robot base, a PKM fixed to the mobile platform, and an adaptable head (Fig. 3).

To accommodate thin-sheet parts with more complex geometry, each mobile agent is equipped with an adaptable head able to conform to the workpiece shape [12]. Shape adaptability (e.g., realized by means of a phase-changing magneto-rheological fluid) together with a reliable adhesion subsystem (e.g., using vacuum suction) allows to quickly achieve a close match of the local geometry of the surface and a secure hold of the part. To ensure good rigidity in

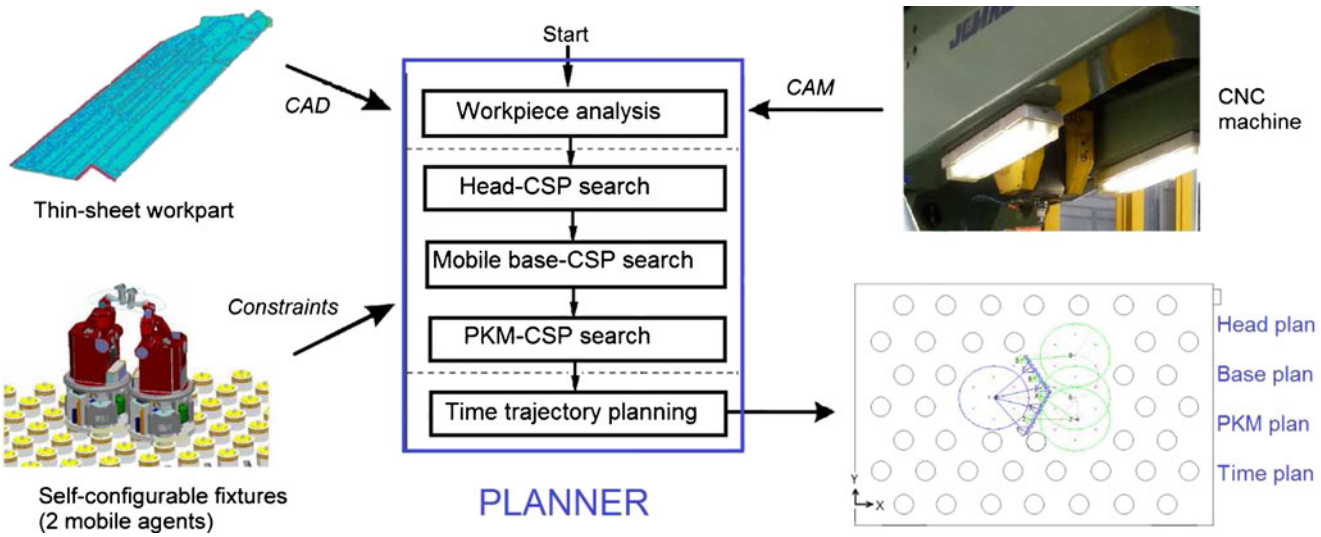

Fig. 2 The general role of the planner in SwarmItFIX

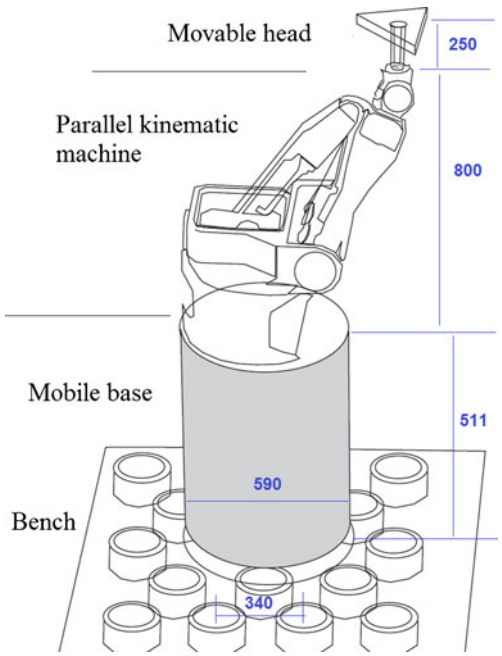

Fig. 3 The bench and the components of a mobile fixturing agent: mobile base, parallel kinematic machine, and adaptable head

all directions, the manipulator used to position and orient the supporting head is a parallel mechanism (or PKM). The SwarmItFIX project uses a new small-scale model of the Exechon tripod, specially designed for this purpose [33]. The manipulator is PLACED on a mobile base capable of rapid locomotion along and secure instant docking to a flat (but not necessarily horizontal) bench [34, 35].

2.2 Work parts

Two aircraft parts, with significantly different sizes and curvatures, were chosen to demonstrate the efficiency of the task planner. Part 1 (Fig. 4) is a left-side subwing fuselage panel. When projected on a plane, its dimensions do not exceed 600 × 700 mm , while its average curvature is 0.0011 mm^{-1}. For part 2, a vertical fin panel, the dimensions and average curvature are 2,800 × 1,100 mm and

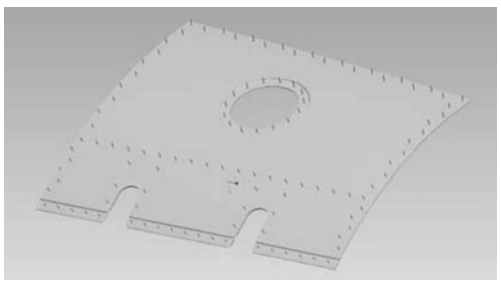

Fig. 4 A regular part with high curvature for hole drilling and milling

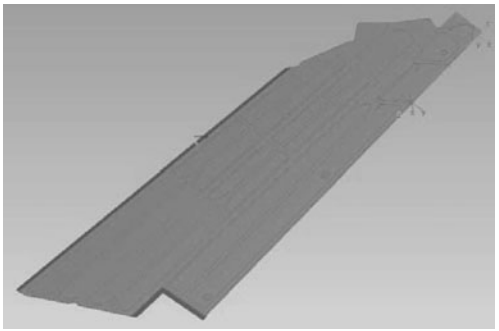

Fig. 5 A nearly flat part with irregular milling contour

0.0003 mm^{-1}, respectively (Fig. 5). These workpieces represent well the range of likely manufacturing scenarios in the production of small to medium airplanes, one of the main targeted application areas of the new fixturing system.

2.3 Planner

Our goal is to develop a task planner for a pair of mobile supporting agents, i.e., to generate the path and time-related (trajectory) plan for such agents. The supporting agents must guarantee stable support for the fragment of the workpiece being currently machined.

The required support stability for a given machining process is achieved when a set of constraints on the relative location of the tool and the supporting heads is satisfied. Moreover, for each agent, the supporting head can move continuously but must remain within the workspace of the PKM, while the mobile bases can only make discrete displacement steps between a finite set of locations, as determined by the docking and locomotion subsystems. Furthermore, motion speed must match the motor constraints.

3 Planner structure

3.1 Modules

The planner is composed of six modules (Fig. 6): an *Init* module, a *control* module (1), three *path-planning* modules (2)–(4), and a *trajectory planner* (5).

The module Init is executed only once per workpiece and its goal is to analyze the workpiece and to decompose it into segments.

Component (1), called *Main Loop*, exercises overall control over plan creation. It may happen that at some contour segment, the on-line execution of the plan must be stopped. This happens when a single plan for the entire workpiece does not exist and the machining process must be split into several parts.

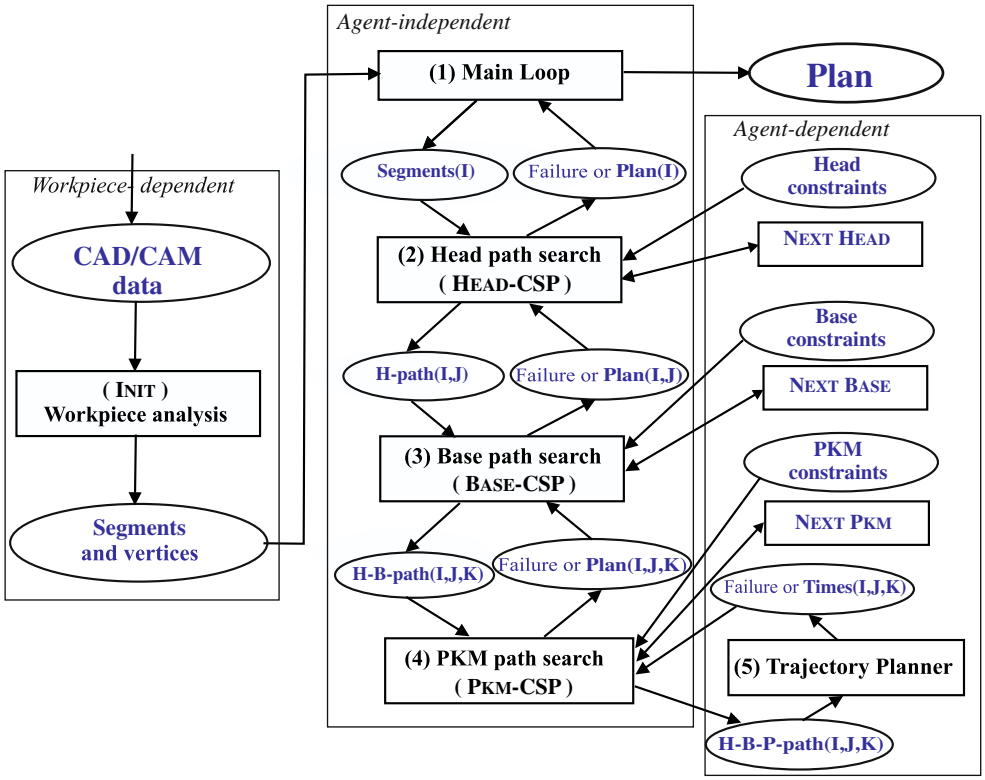

Fig. 6 The modular structure of the planner with its three main parts: workpiece dependent, agent independent, and agent dependent. The counters I, J, and K denote the incremental creation and verification of the path plan. Index *I* denotes the currently planned part of the contour, *J* counts a segment within the current given contour part, and *K* indicates the position along the current segment

The remaining four modules constitute consecutive layers in a hierarchic structure. The modules *Head-CSP* (2), *Base-CSP* (3), and *PKM-CSP* (4) perform appropriate stages of path planning while using a so-called Triple-CSP approach. By applying an incremental state-space search, these subroutines create head paths, base paths, and PKM paths.

After a path plan is found (by modules (2)–(4)), it is verified by module (5), the *Trajectory Planner*. All actions of the path plan must be executed in time and in a given order defined by situation indices. The Trajectory Planner tests if the obtained path plan satisfies the constraints resulting from the time limits (set by the task requirements and the system's dynamic capabilities).

3.2 The Init module

This module performs workpiece analysis by considering the head parameters, the CAD/CAM data describing the workpiece, and the contour to be machined. Before path planning begins, the contour is divided into line segments separated by vertices. The task of Init is to identify and classify these vertices. The distinguished classes depend on the head shape; we use an equilateral triangular head with 100-mm sides. In general, a contour *segment* is a polyline with many intermediate line endings and two segment endings called *vertices*. In practice, we expect the length of a line segment to be at least 100 mm (the side length of the head).

A contour point is a true *vertex* if it satisfies the following conditions:

1. It marks a line segment ending, i.e., a point at which a rapid change of the contour's curvature appears, measured within a local neighborhood.
2. Two consecutive vertices should be connected by a sufficiently long line segment (this condition is needed to avoid contour segments with lengths less than a head's side length—if two line ending points are at close distance to each other, then only one of them can be a vertex point).

The vertex type is determined by the angle value, α, between the two adjoining line segments. For the chosen triangular head, these types are defined (Fig. 7):

- Type 0: $\alpha < 60°$, $\rightarrow$ a start vertex;
- Type 1: $60° \leq \alpha \leq 90°$, $\rightarrow$ a vertex populated by a single head with two active sides;

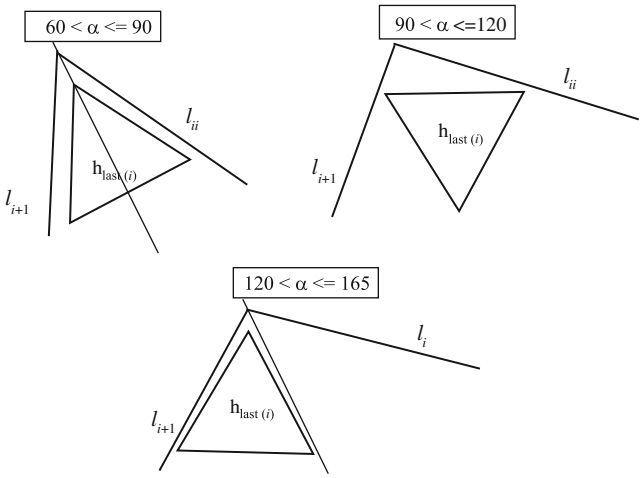

Fig. 7 Positioning the last head assigned to a line segment l_i at a vertex of type 1, 2, or 3

- Type 2: $90° < \alpha \leq 120°$, → a vertex populated by a single head with one active side;
- Type 3: $120° < \alpha \leq 165°$, → a vertex populated by two heads.

The line ending points with aligned angles, $\alpha > 165°$, are considered only to be endpoints of inner-segment edges (and not true segment vertices).

3.3 Main loop

This module performs the overall control of the path-planning process. The *Head Planner* (module (2)) is iteratively called for subparts of the contour if no single plan for the entire contour can be generated. Eventually, each call returns a partial plan. Partial plans are combined into a total plan. The handling of eventual failures returned by module (2) is as follows: changing the start vertex and/or splitting the contour into subcontours for which independent plans need to be found.

The input to this module is a contour description consisting of a sequence of classified line segments and corresponding vertices. At the Ith iteration, it calls Head Planner by passing *Segments(I)* (data describing the entire contour or a subset of it). Head Planner either returns a *Failure* (then the current vertex is marked as "failed") or success with a candidate path *Plan*.

The Main Loop module combines eventual partial plans obtained by iterative calls of the three CSP modules into a complete plan for the entire contour.

3.4 Triple-CSP for cooperating mobile agents

Formally, a discrete CSP means a search for assignments to a finite set of variables: $\{\mathbf{X}_i = \mathbf{d}_i | i = 1, ..., N\}$, where $\mathbf{X}_i$ and $\mathbf{d}_i \in \mathbf{D}_i$ are variables and assigned values, respectively.

A solution to a problem represented as a CSP is found by a (partly application independent) depth-first search with backtracking in the decision tree of alternative assignments to give problem variables. A solution to the CSP is every complete assignment (i.e., a path from root node to terminal node of length N) that satisfies the set of constraints. With such a model of the problem, an application-independent reusable search algorithm can be designed. Then, it is sufficient to fit the search-based control with some agent-dependent data and functions—related to single-variable assignments and constraints—to design a complete solution algorithm.

3.4.1 CSP variables

Let us distinguish three Cartesian coordinate systems, with respect to which the agent's parameter values will be set:

1. W—the world coordinate system $X^{\mathrm{W}}Y^{\mathrm{W}}Z^{\mathrm{W}}$ affixed to a selected corner of the bench,
2. B—the mobile base coordinate system, a local system affixed to the center of the base,
3. P—the local PKM coordinate system $X^{\mathrm{P}}Y^{\mathrm{P}}Z^{\mathrm{P}}$ affixed to the center of the PKM end-effector platform.

Appropriate parameter vectors will be defined with respect to one of these systems, for every agent's part: base, head, and PKM. The parameter vectors are defined in Section 3.4.2. A separate head coordinate system is not referred—the head variables will be expressed in the global coordinate system. The world (bench) coordinates are related to the CNC coordinates by a simple translation in the X–Z plane.

As the required plan will contain a sequence of positions in time for every agent, the parameter vector of every part needs to be extended by two discrete situation indices:

- T_{beg}—time index of the latest possible action end to reach the current position,
- T_{end}—time index of the earliest possible action start to move to the next position.

The plan contains three lists of states for the subcomponents of each agent, containing the consecutive locations of heads, bases, and PKMs. A transition between two consecutive states of one agent is specified as follows:

1. Head transition → by a sequence of PKM states;
2. Base transition and corresponding rotation of PKM platform → by a "base action" state $\mathbf{b}_{\mathrm{action}}$;
3. PKM, no explicit transition specification—we assume that the transitions between two consecutive PKM states are performed linearly in the object space.

3.4.2 State parameters

The state of the *head* part (Appendix 1) of a single agent is as follows:

$$\mathbf{h} = \left[T_{\text{beg}}, T_{\text{end}}, \ {}_{\text{h}}c_x^{\text{W}}, \ {}_{\text{h}}c_y^{\text{W}}, \ {}_{\text{h}}c_z^{\text{W}}, \ {}_{\text{h}}\alpha^{\text{W}}, \ {}_{\text{h}}\beta^{\text{W}}, \ {}_{\text{h}}\gamma^{\text{W}}\right], \tag{1}$$

where $({}_{\text{h}}c_x^{\text{W}}, \ {}_{\text{h}}c_y^{\text{W}}, \ {}_{\text{h}}c_z^{\text{W}})$ is the location of the head reference point in world coordinates (i.e., relative to bench) and ${}_{\text{h}}\alpha^{\text{W}}$, ${}_{\text{h}}\beta^{\text{W}}$, and ${}_{\text{h}}\gamma^{\text{W}}$ are rotation angles (around Z, Y, X, respectively) specifying the head orientation in world frame. The *actions* of a head are due to the transitions of the PKM state: every head structure contains a list of PKM state indices—these PKM states are necessary to reach a given head state from the predecessor head state.

The state of the *base* part of a single agent is as follows:

$$\mathbf{b} = \left[T_{\text{beg}}, T_{\text{end}}, \ {}_{\text{b}}c_x^{\text{W}}, \ {}_{\text{b}}c_y^{\text{W}}, \ {}_{\text{b}}c_z^{\text{W}}, \ {}_{\text{b}}\theta^{\text{W}}; {}_{\text{b}}\, pkm\theta^{\text{W}}\right], \tag{2}$$

where ${}_{\text{b}}c_x^{\text{W}}$, ${}_{\text{b}}c_y^{\text{W}}$, and ${}_{\text{b}}c_z^{\text{W}}$) are the 3D world coordinates of the base's reference point ${}_{\text{b}}C$ (i.e., position relative to the bench plane origin); ${}_{\text{b}}\theta^{\text{W}}$ is the rotation angle, which specifies the orientation of base in the 2D coordinate system $X^{\text{W}}Z^{\text{W}}$, related to the bench plane; and ${}_{\text{b}}pkm\theta^{\text{W}}$ specifies the orientation of the rotatable PKM platform, also given in the global 2D coordinate system $X^{\text{W}}Z^{\text{W}}$.

We consider ${}_{\text{b}}pkm\theta^{\text{W}}$ to be an element of the base state, to check a constraint with respect to ${}_{\text{b}}\theta^{\text{W}}$ (the orientation difference ${}_{\text{b}}\theta^{\text{W}} - {}_{\text{b}}\, pkm\theta^{\text{W}}$ must be within $< -2\pi, 2\pi >$) by the Base-CSP search module.

In order to transit from the current base position to the next base position, there is a list of *base-action* states specified, where a single *base action* is as follows:

$$\mathbf{b}_{\text{action}} = \left[T_{\text{beg}}, T_{\text{end}}, \ {}_{\text{b}}pin^{\text{B}}, \ {}_{\text{b}}d\theta^{\text{W}}, \ {}_{\text{b}}dPkm\theta^{\text{W}}\right]. \tag{3}$$

Here, we need to determine the *pin* around which the rotation will be done, the amount and sign of this rotation $({}_{\text{b}}d\theta^{\text{W}})$, and also the (usually contradicting) rotation of the PKM platform $({}_{\text{b}}dPkm\theta^{\text{W}})$.

The state of the *PKM* of an agent is as follows:

$$\mathbf{p} = \left[T_{\text{beg}}, T_{\text{end}}, \phi_0^{\text{W}}, leg_1, leg_2, leg_3, \psi_1^{\text{P}}, \psi_2^{\text{P}}, \psi_3^{\text{P}}, \beta 7^{\text{P}}\right], \tag{4}$$

where ϕ_0^{B} is the rotation angle of frame XY attached to the PKM platform (the PKM frame), expressed in world frame; (leg_1, leg_2, leg_3) are the lengths of three legs; and ψ_1^{B}, ψ_2^{B}, and ψ_3^{B}) are the rotation angles of the PKM wrist and these are the Z–Y–Z Euler angles expressed in PKM frame. The final seventh degree of freedom β_7^{B} is a rotation angle around the current head's Z axis, expressed in the PKM frame.

A PKM action, which specifies how to transit from one PKM state to the other, is defined by the agent's controller itself. In the path plan, we need only to specify a collision-free sequence of PKM states that leads from the current head position to the next one. This solution is found during PKM-CSP search.

3.4.3 Constraints

A path plan for the agents needs to satisfy a set of geometric constraints. For the cooperating mobile agents, we need to define the following:

- Geometric constraints between agents and the workpiece contour: expressing the necessary physical requirements for adequate support for the given workpiece and machining process,
- The workspace of the PKM: used to check quickly feasible base–head position pairs,
- Geometric constraints between bases and PKMs: needed to avoid collisions between agents during base position transitions,
- The inverse kinematics problem solution of the PKM: used when defining feasible PKM states for consecutive head positions.

4 Time (trajectory) planner

This module requires dynamic models of the bases, the PKMs, and the heads, so that the durations of various agent actions can be evaluated, e.g., locking and unlocking of the bases, accelerating and decelerating a PKM leg motion, and mounting and demounting the head on the workpiece.

The time schedule of an agent's actions is induced by the required time points of head support and head relocation. As illustrated in Fig. 8, there is basically a sequence of three time periods that appears repeatedly. In the first period, $[T_{\text{beg1}}, T_{\text{beg2}}]$, only the first agent is supporting the workpiece and its state $h1$ is constant; in the second period, $[T_{\text{beg2}}, T_{\text{end1}}]$, both agents need to be fixed and jointly support the workpiece; and finally in the third period, $[T_{\text{end1}}, T_{\text{beg3}}]$, the second agent's state $h2$ is constant, whereas the first agent is transitioning from $h1$ to $h3$.

The limits of these time intervals are crucial for time planning. Figure 9 illustrates how they are obtained. Based on the support-force distribution analysis, we can set a field threshold F for single head support. This leads to the constraint, active_field $\leq F$, where active_field is associated with the active head side. For every head location, equality is reached for two contour points, $k1$ and $k2$.

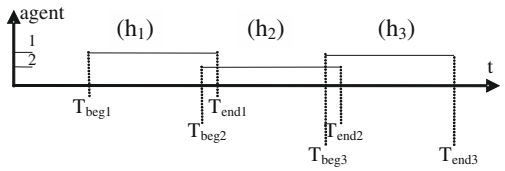

Fig. 8 The basic sequence of three time intervals

The values of $k1$ and $k2$ for a head location determine two of the time-interval endpoints for the neighboring head locations. Indeed, assume that the tool moves along the contour from left to right and from top to bottom. When it enters the active field zone (where the active_field constraint is satisfied) at the point $k1$, it sets the time point $T_{\text{end}-\text{prev}}$. When it leaves, the tool sets the time point $T_{\text{beg}-\text{next}}$. Here, prev and next refer to the previous and next head states in the plan. During the period, $[T_{\text{beg}-h1}, T_{\text{end}-h1}]$, the head $h1$ should stay in a constant position, while the second head is moving to its new state. In practice, the head $h1$ should be fixed in a given position not later than $T_{\text{beg}-h1}$ and it should start its relocation action not earlier than $T_{\text{end}-h1}$. In corner areas of the workpiece, a head usually has two active support sides. In such a case, the inner limit points of the two fields are not relevant—the tool enters the left side of the first field and leaves the right side of the second field.

5 Results

Test of plans generated for various drilling and milling processes have been run in real-life conditions (Fig. 10). In the following, two examples of successfully executed plans are given.

5.1 Discrete domains

In general, the state variables for the heads, the mobile bases and the PKMs, are real-valued vectors. This would lead to an infinite number of possible positions and configurations of the agent. But a discrete CSP can efficiently handle only a finite state space. Therefore, the continuous domain of each variable needs to be approximated by a finite set of discrete values taken with sufficient resolution. For example, if a

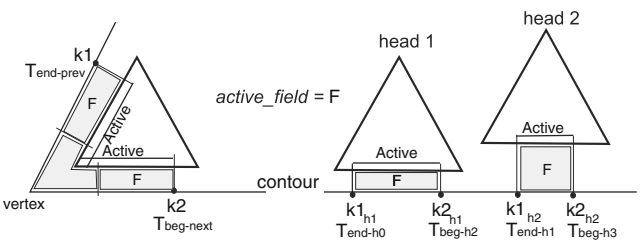

Fig. 9 The main time points related to "support field" constraints

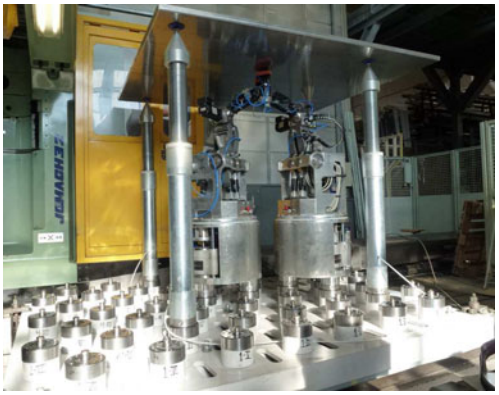

Fig. 10 The real arrangement of the system (the bench and two mobile agents) in a factory

parameter representing a rotation angle can take values from a continuous interval, $[-30°, 30°]$, then we might be satisfied with an approximation with a resolution of 1° by a set of 61 discrete angle values: $\{-30°, -29°, ..., 0°, ..., 29°, 30°\}$. The resolution of relative discrete locations for a single head can be set to $n \times n$, where $n = 1 + (2d_{\max}/d_{\min})$.

Even with such a necessary discrete realization, the number of domain values can be very high. Fortunately, for most of our CSP variables, there exist local optimization criteria that allow to order the domain values starting with the most promising ones. For example, looking for a relative orientation of the head triangle along a workpiece contour, we shall select the parallel-to-contour orientation (i.e., with an angle value of 0°) as the best one.

5.2 Regularity conditions

The planner can return a feasible path plan if the workpiece contour is *regular*, i.e., if it satisfies these properties:

1. Line regularity: For each segment, at least one head can be placed between the adjoining vertices.
2. Area regularity: The inner area bordering three consecutive edges of a contour is sufficiently large to include two consecutive heads.

These are necessary properties—required by our head allocation strategy. A single segment needs to be supported by at least one head position (causing line regularity) and a single head can have up to two "active" sides (this explains the area regularity). In general, there is no guarantee that a feasible plan will be found for regular contours.

5.3 Workpiece segmentation

The segmentation results for the contour of part 2 (Fig. 5) are illustrated in Fig. 11.

Contour corners are not considered to be vertices (i.e., endpoints of segments) if their angles are larger than 120°

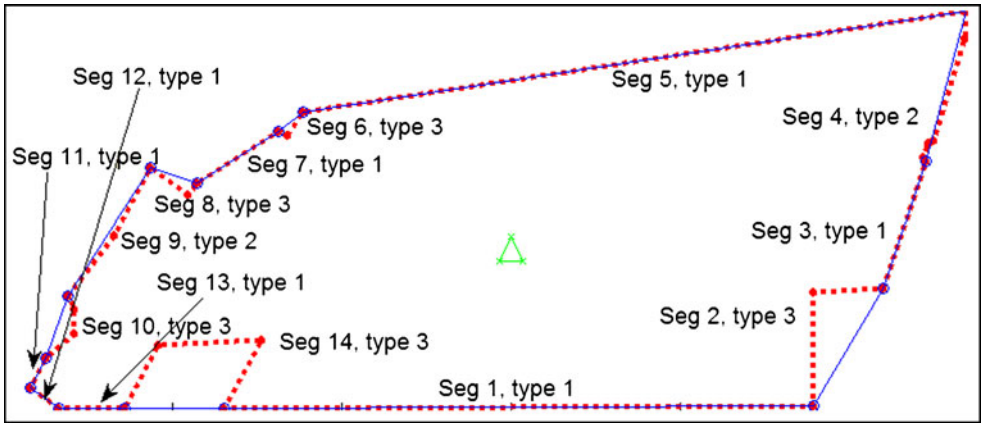

Fig. 11 The appropriate start vertex is selected and the segments are renumbered. In this case, the vertex of type 0 is selected as the start vertex

or they are at a small distance (as compared to the head side length) to a "dominating" corner (i.e., with a lower vertex-type index). Hence, some more corners are filtered from the vertices list in order to avoid small-size segments.

The start point is selected at a vertex of type 0. It is difficult or even impossible to locate any intermediate head at such a corner, and so the path plan would normally be interrupted there. In contrast, a type 0 "start vertex" causes no problems.

The contour of part 1 (Fig. 4) consists mainly of long linear segments causing no serious difficulties for the head initialization procedure. The part 2 contour is more difficult to process as it consists of many small line segments and the contour's direction changes rapidly many times.

5.4 Head and base path plan

Consider the milling process of the contour of part 2 first. The head distance thresholds are set as follows: $D_{\min} = 2$ mm, $D_{\max} = 20$ mm, and $D_{\mathrm{h}} = 20$ mm. Many candidate head path plans satisfying the head constraints can be generated.

The parameters used by Base-CSP correspond to the bench resolution and the mobile base size: distance between pins (340.35 mm), working zone radius (between 340 and 460 mm), and base perimeter (590 mm). An illustration of the head and base path plans with two mobile agents for a considered workpiece is shown in Fig. 12.

5.5 Drilling around a circle

We apply the same approach for the path plan generation of a hole-drilling process (Fig. 13). There will be a difference only at the time-related (trajectory) planning stage. Let us consider two hole-drilling sequences required for the second contour: (a) hole drilling around a circle and (b) along a polygonal-like contour.

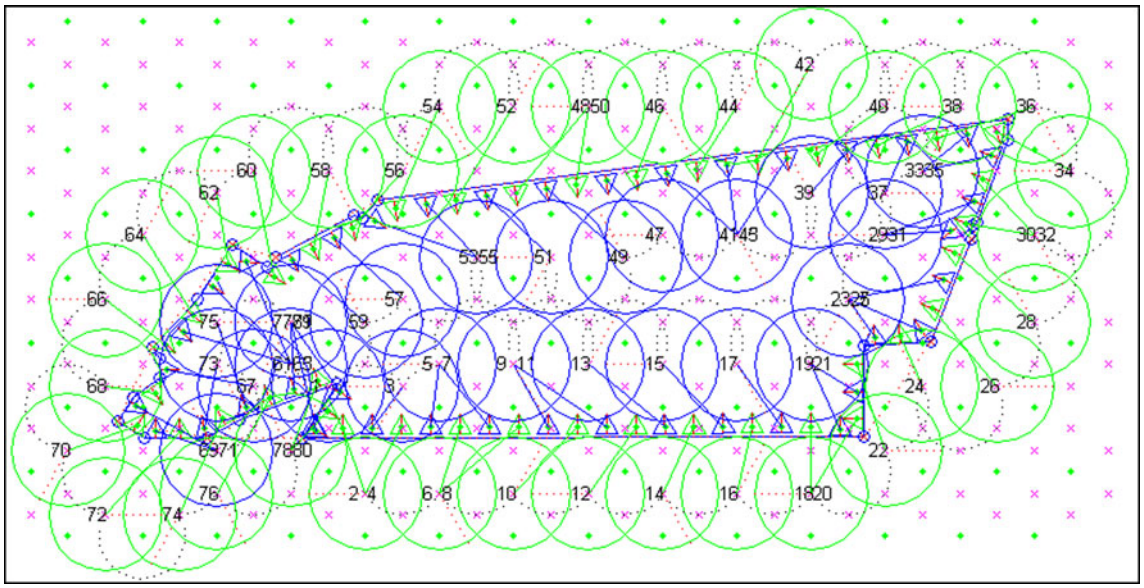

Fig. 12 The head and base plans for the entire contour (*top view*—projection onto the bench)

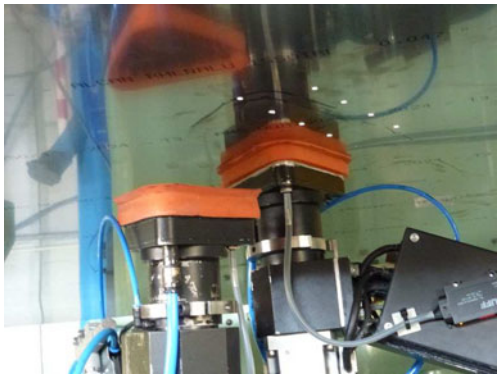

Fig. 13 Illustration of actual mobile support during hole drilling in a factory

Our approach requires that the circle is first approximated by a regular polygon. There can be many possible approximations with different number of sides. A side must be sufficiently long, at least equal to the half of the head side length.

There are seven holes to be drilled; hence, let us first approximate the circle by a seven-sided open polyline, where the lines are of tangential directions to the circle at the drilling points. The edge lengths are similar to a head side length. For a second approximation, we use a 15-sided closed polygon. In both cases, the planner recognizes the contour to consist of a single segment only. The head plans have seven and eight head states (Fig. 14). The head and base plan for a 15-sided polygon is shown on Fig. 15.

5.6 Drilling along a contour

A complete plan consists of four parts, i.e., the path plans for heads, bases, and PKMs, as well as the time plan for all positions and actions in the path plans. Let us consider the hole-drilling process along the outer contour of the first workpiece (part 1). In Fig. 16, we illustrate the head and base path plans generated for the first part of this drilling process. For the corresponding PKM path plan, given in XML format, see Appendix 2. The time (trajectory) plan for

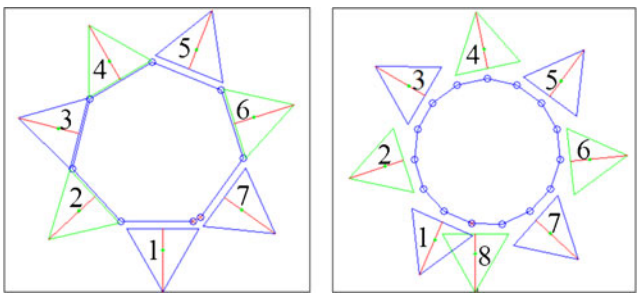

Fig. 14 Two head plans for the hole drilling around a circle approximated by a 7- or 15-sided-polygon

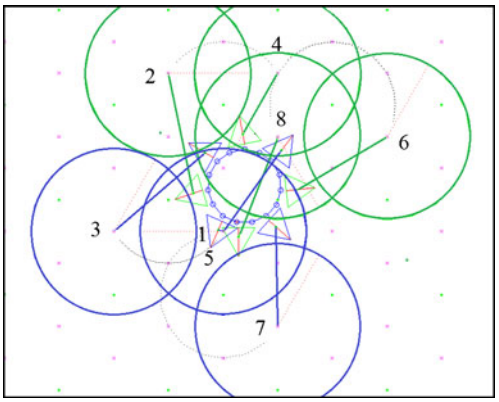

Fig. 15 The head and base plans for the hole drilling around a circle approximated by a 15-sided-polygon

a drilling or milling is specified in terms of the <TBeg> and <TEnd> fields (i.e., the enter end exit times) for every part (head, base, PKM) of every agent. In the experiment, the tool speed along distances between holes is 5 mm/s, while the total time for drilling a single hole is 3 s. The plan contains also a time schedule for visiting contour corners (in milling) and holes (in drilling) (both <TBeg> and <TEnd> times) that allows the agent's controller [32] to synchronize the plan execution with the CNC machine and to react to unexpected events or time delays.

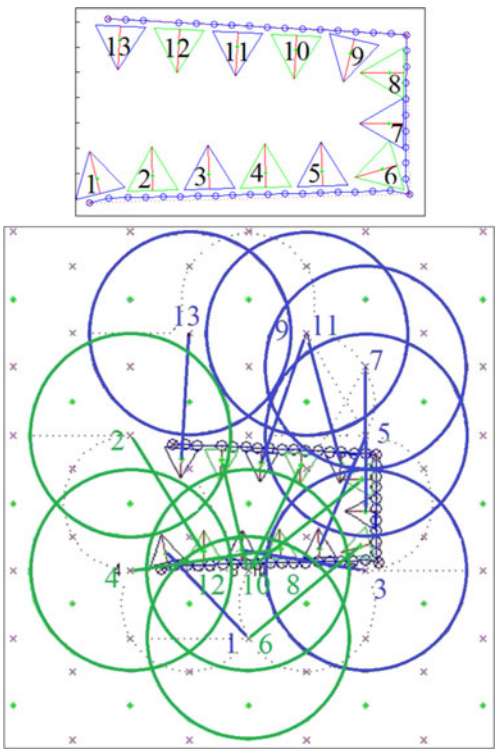

Fig. 16 Head and base plans for hole drilling along the outer contour of the first workpiece

6 Conclusions

Our work addresses a reconfigurable robotic fixture autonomously offering stable support during the machining of complex and flexible parts. This novel industrial solution to the fixturing problems can provide a valuable highly automated component in the organization of production and, in this way, contribute to the further development of the flexible and resilient manufacturing systems of the future [9].

The originality of our contribution stems from the novelty of the SwarmItFIX concept. The fixturing system has an unprecedented degree of autonomy and the capacity to handle a wide variety of parts, including large and flexible thin sheets. Other existing systems still require human intervention in order to perform reconfiguration, and only very few are able to support flexible parts [37–41]. The new system has significant advantages over state-of-the-art RFAs or pin-type systems; it has the potential to be a more agile, smarter, and cheaper industrial solution.

The new concept of fixturing poses a novel and multi-faceted task-planning problem, resolved by the CSP planner. It follows the methodology of CSPs. In our approach, we define the path-planning problem for the three parts of agents (head, mobile base, and PKM) in terms of three CSPs. Thus, the same computation tool, an incremental search algorithm for CSP, can be used to solve the three parts of the planning problem. For efficiency reasons, a hierarchy of three incrementally solved CSPs is proposed. This structure allows to verify single assignments within Head-CSP, performed for the head plan, by assignments within the Base-CSP, and these in turn are verified by assignments within the PKM-CSP.

An important design feature is the use of local optimization as a search heuristics, i.e., the competitive variable assignments are ordered according to appropriate optimization criteria and "better" assignments are selected and checked first. The general search approach can be applied in a variety of related applications: one needs only to provide the search-based algorithm with appropriate agent-dependent and process-dependent data and to add functions related to single-variable assignments and constraints.

Acknowledgments This work was supported by the European Union Framework Program Theme [NMP-2007-3.2-1] within the Project SwarmItFIX, grant no. FP7-214678.

Appendix 1: Notation

$\mathbf{h}_i$	Head state (a sequence of head states is assigned to a variable in "Head-CSP" search)
$\mathbf{b}_i$	Mobile base state (assigned to a variable in "Base-CSP" search)
$\mathbf{p}_i$	PKM state (assigned to a variable in "PKM-CSP" search)
$\mathbf{S}$	Agent's state, $\mathbf{S} = [\mathbf{h}, \mathbf{b}, \mathbf{p}]$
X_i	Variable in general CSP search
X_i^{H}	Variable in Head-CSP search
X_i^{B}	Variable in Base-CSP search
X_i^{P}	Variable in PKM-CSP search
$D_{\min}$	Distance threshold for head location vs. contour
$D_{\max}$	Distance threshold for head location vs. contour
D_{h}	Distance threshold for maximum distance between two successive heads
T_{beg}	Time index of latest possible action end to reach the current position
T_{end}	Time index of earliest possible action start to move to the next position

Appendix 2: PKM plan

Illustration of the PKM plan in the hole-drilling example:

```
<pkm>
    <item>
       <ind>0</ind> <alpha0>1.8186</alpha0>
       <l1>0.328078</l1> <l2>0.304345</l2>
       <l3>0.329079</l3>
       <psi1>0.0871006</psi1> <psi2>-1.05742</psi2>
       <psi3>-0.137281</psi3>
       <beta7>-0.526275</beta7>
       <agent>1</agent>
       <TBeg>3</TBeg> <TEnd>19.2926</TEnd>
    </item>
    <item>
       <ind>100</ind> <alpha0>4.63124</alpha0>
       <l1>0.301924</l1> <l2>0.287641</l2>
       <l3>0.301287</l3>
       <psi1>0.0153379</psi1> <psi2>-0.955512</psi2>
       <psi3>-0.0232599</psi3>
       <beta7>0.630085</beta7>
       <agent>2</agent>
       <TBeg>17.7844</TBeg> <TEnd>51.9366</TEnd>
    </item>
```

```
  <item>
     <ind>120</ind> <alpha0>1.8186</alpha0>
     <l1>0.300895</l1> <l2>0.275811</l2>
     <l3>0.302872</l3>
     <psi1>0.0880273</psi1> <psi2>-1.07176</psi2>
     <psi3>-0.140693</psi3>
     <beta7>-0.526275</beta7>
     <agent>1</agent>
     <TBeg>19.979</TBeg> <TEnd>19.979</TEnd>
  </item>
  <item>
     <ind>140</ind> <alpha0>1.98294</alpha0>
     <l1>0.32085</l1> <l2>0.24162</l2>
     <l3>0.32085</l3>
     <psi1>0</psi1> <psi2>-1.44318</psi2>
     <psi3>0</psi3>
     <beta7>-0.690617</beta7>
     <agent>1</agent>
     <TBeg>20.2024</TBeg> <TEnd>20.2024</TEnd>
  </item>
  <item>
     <ind>160</ind> <alpha0>2.49461</alpha0>
     <l1>0.32085</l1> <l2>0.24162</l2>
     <l3>0.32085</l3>
     <psi1>0</psi1> <psi2>-1.44318</psi2>
     <psi3>0</psi3>
     <beta7>-1.42207</beta7>
     <agent>1</agent>
     <TBeg>21.3167</TBeg> <TEnd>21.3167</TEnd>
  </item>
  <item>
     <ind>180</ind> <alpha0>2.33026</alpha0>
     <l1>0.267749</l1> <l2>0.301251</l2>
     <l3>0.267488</l3>
     <psi1>0.0081074</psi1> <psi2>-0.681787</psi2>
     <psi3>-0.0109748</psi3>
     <beta7>-1.25772</beta7>
     <agent>1</agent>
     <TBeg>21.5516</TBeg> <TEnd>21.5516</TEnd>
  </item>
  <item>
     <ind>200</ind> <alpha0>2.33026</alpha0>
     <l1>0.295595</l1> <l2>0.326589</l2>
     <l3>0.295349</l3>
     <psi1>0.00778439</psi1> <psi2>-0.689414</psi2>
     <psi3>-0.0105845</psi3>
     <beta7>-1.25772</beta7>
     <agent>1</agent>
     <TBeg>22.2374</TBeg> <TEnd>22.2374</TEnd>
  </item>
</pkm>
  <pNum>57</pNum>
```

```
<pkm>
      <item>
         <ind>220</ind> <alpha0>4.63124</alpha0>
         <l1>0.274351</l1> <l2>0.259441</l2>
         <l3>0.273685</l3>
         <psi1>0.0156016</psi1> <psi2>-0.96785</psi2>
         <psi3>-0.0239012</psi3>
         <beta7>0.630085</beta7>
         <agent>2</agent>
         <TBeg>52.6226</TBeg> <TEnd>52.6226</TEnd>
      </item>
...
      <item>
         <ind>300</ind>   <alpha0>4.09309</alpha0>
         <l1>0.291838</l1> <l2>0.319597</l2>
         <l3>0.292371</l3>
         <psi1>-0.0134551</psi1> <psi2>-0.700625</psi2>
         <psi3>0.0183708</psi3>
         <beta7>-0.926153</beta7>
         <agent>2</agent>
         <TBeg>54.9178</TBeg> <TEnd>54.9178</TEnd>
      </item>
...
 </pkm>
   <pNum>57</pNum>
```

References

1. Menassa R, De Vries W (1991) Optimization methods applied to selecting support positions in fixture design. ASME J Eng Ind 113(4):412–418
2. Cai W, Hu S, Yuan J (1996) Deformable sheet metal fixturing: principles, algorithms, and simulations. J Manuf Sci Eng Trans ASME 118(3):318–324
3. Vallapuzha S, De Meter E, Choudhuri S, Khetan R (2002) An investigation of the effectiveness of fixture layout optimization methods. Int J Mach Tool Manuf 42(2):251–263
4. Sela M, Gaudry O, Dombre E, Benhabib B (1997) A reconfigurable modular fixturing system for thin-walled flexible objects. Int J Adv Manuf Technol 13(9):611–617
5. Shirinzadeh B, Tie Y (1995) Experimental investigation of the performance of a reconfigurable fixture system. Int J Adv Manuf Technol 10(5):330–341
6. Youcef-Toumi K, Liu W, Asada H (1988) Computer-aided analysis of reconfigurable fixtures and sheet metal parts for robotics drilling. Robot Comput Integr Manuf 4(3–4):387–393
7. Bi Z, Zhang W (2001) Flexible fixture design and automation: review, issues and future directions. Int J Prod Res 39(13):2867–2894
8. Kang Y, Rong Y, Yang J, Ma W (2002) Computer-aided fixture design verification. Assem Autom 22(4):350–359

9. Zhang WJ, van Luttervelt CA (2011) Toward a resilient manufacturing system. CIRP Ann Manuf Technol 60:469–472
10. Boyle I, Rong Y, Brown D (2011) A review and analysis of current computer-aided fixture design approaches. Robot Comput Integr Manuf 27(1):1–12
11. Molfino R, Zoppi M, Zlatanov D (2009) Reconfigurable swarm fixtures. In: REMAR 2009. Proceedings of the 2009 ASME/IFToMM international conference on reconfigurable mechanisms and robots, vol IEEE : CFP0943G-PRT, pp 730–735
12. de Leonardo Girard LM, Zoppi M, Li X, Zlatanov D, Molfino RM (2013) SwarmItFIX: a multi-robot-based reconfigurable fixture. Ind Robot Int J 40(4):320–328
13. Gonzalez-Rodriguez A, Gonzalez-Rodriguez A (2011) Collision-free motion planning and scheduling. Robot Comput Integr Manuf 27(3):657–665
14. DeMeter E (1998) Fast support layout optimization. Int J Mach Tool Manuf 38(10-11):1221–1239
15. Asante JN (2010) Effect of fixture compliance and cutting conditions on workpiece stability. Int J Adv Manuf Technol 48(1–4):33–43
16. Papastathis TN, Ratchev SM, Popov AA (2012) Dynamics model of active fixturing systems for thin-walled parts under moving loads. Int J Adv Manuf Technol 62(9-12):1233–1247
17. Rearick M, Hu S, Wu S (1993) Optimal fixture design for deformable sheet metal workpieces. Trans NAMRI/SME 21:407–412
18. Canny J (1993) Computing roadmaps of general semi-algebraic sets. Comput J 36(5):504–514
19. Choset H, Lynch K, Hutchinson S, Kantor G, Burgard W, Kavraki L, Thrun S (2005) Principles of robot motion: theory, algorithms, and implementations. MIT Press, Cambridge
20. LaValle S (2006) Planning algorithms. Cambridge University Press, Cambridge
21. Kavraki LE, Svestka P, Latombe J-C, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 12(4):566–580
22. Kuffner JJ, LaValle SM (2000) RRT-connect: an efficient approach to single-query path planning. In: IEEE international conference on robotics and automation. San Francisco, pp 995–1001
23. Jaillet L, Cortes J, Simeon T (2010) Sampling-based path planning on configuration-space costmaps. IEEE Trans Robot 26(4):635–646
24. Brock O, Khatib O (2002) Elastic strips: a framework for motion generation in human environments. Int J Robot Res 18(6):1031–1052
25. Latombe J-C (1991) Robot motion planning. Kluwer, Boston
26. Ratliff N, Zucker M, Bagnell JA, Srinivasa S (2009) Chomp: gradient optimization techniques for efficient motion planning. In: IEEE international conference on robotics and automation (ICRA), pp 489–494
27. Russel S, Norvig P (2002) Artificial intelligence. A modern approach. Prentice-Hall, Upper Saddle River
28. Brailsford SC, Potts CN, Smith BM (1999) Constraint satisfaction problems: algorithms and applications. Eur J Oper Res 119(3):557–581
29. Apt KR (2003) Principles of constraint programming. Cambridge University Press, Cambridge
30. Bartak R, Salido MA, Rossi F (2010) Constraint satisfaction techniques in planning and scheduling. J Intell Manuf 21(1):5–15
31. Sun Z, Zhang B, Cheng L, Zhang WJ (2011) Application of the redundant servomotor approach to design of path generator with dynamic performance improvement. Mech Mach Theory 46:1784–1795
32. Zieliński C, Kornuta T, Trojanek P, Winiarski T, Walecki M (2012) Specification of a robot-based reconfigurable fixture control system. In: Robot motion and control 2011. Lecture notes in control and information sciences, vol 422. Springer, London, pp 171–182
33. Neumann K-E (2008) Structure concept of Exechon PKM, Tech. Rep. 34, Exechon, Stockholm. exechonworld.com/document/200804/article34.htm. Accessed 8 July 2013
34. Li X, Zoppi M, Molfino R, de Leonardo Girard LM (2011) Design of mobile base for a self-reconfigurable intelligent swarm fixture system. In: CLAWAR 2011. The 14th international conference on climbing and walking robots and the support technologies for mobile machines, Paris, 6-8 September 2011, pp 925–932
35. Zoppi M, Molfino R, Zlatanov D (2011) Bench and method for the support and manufacturing of parts with complex geometry. European Patent WO2011EP58992 20110531
36. Szynkiewicz W, Zielińska T, Kasprzak W (2010) Robotized machining of big work pieces: localization of supporting heads. Front Mech Eng China 5(4):357–369. http://springerlink.com/content/1673-3479/5/4/
37. Arzanpour S, Fung J, Mills JK, Cleghorn WL (2006) Flexible fixture design with applications to assembly of sheet metal automotive body parts. Assem Autom 26(2):143–153
38. Shen CH, Lin YT, Agapiou JS, Bandyopadhyay P (2006) Reconfigurable fixtures for automotive engine machining and assembly applications. In: Reconfigurable manufacturing systems and transformable factories. Springer, Berlin, pp 155–194
39. Bi Z, Lang S, Verner M, Orban P (2008) Development of reconfigurable machines. Int J Adv Manuf Technol 39(11):1227–1251
40. Jonsson M, Ossbahr G (2010) Aspects of reconfigurable and flexible fixtures. Prod Eng 4(4):333–339
41. Papastathis T, Bakker O, Ratchev S, Popov A (2012) Design methodology for mechatronic active fixtures with movable clamps. Procedia CIRP 3:323–328