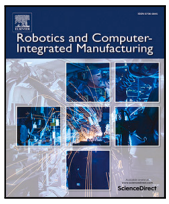
A unified sampling method for optimal feature coverage and robot placement

Robotics and Computer–Integrated Manufacturing 93 (2025) 102932

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim



Full length Article

A unified sampling method for optimal feature coverage and robot placement

Domenico Spensieri *, Edvin Åblad, Raad Salman, Johan S. Carlson

Geometry and Motion Planning Group, Fraunhofer-Chalmers Centre, Chalmers Science Park, Gothenburg, 41288, Sweden

ARTICLE INFO

Keywords:
Robot cell design
Inspection coverage
Optimal placement
Collision avoidance

ABSTRACT

Designing a robot line includes the critical decision about the number of robots needed to carry out all the tasks in the stations and their placement. Similarly, having a robot manipulator mounted on a mobile base, such as an Automated Guided Vehicle (AGV), needs a careful choice of the base positions to minimize cycle time for the operations. In this paper, we solve both the robot placement and the AGV positioning problems by relating them to feature coverage applications, where the challenge is to place cameras (or other sensors) to inspect all points on a workpiece for metrology tasks. These similarities allow us to design an efficient divide&conquer-based algorithm which can be adapted to solve all three problems above, where finding the minimum number of positions for sensors, AGVs and robots is crucial to reduce cycle time and costs.

The algorithm is divided in two parts: the first one is responsible for identifying candidate positions, whereas the second solves a set covering problem. We show that these two parts can even be interlaced to obtain high-quality solutions in short time.

A successful computational study has been carried out with both artificial instances and three industrial scenarios, ranging from laser sensor inspection cells in the aerospace industry, to an automated cleaning room, and ending with a stud welding station for automotive applications.

The results show that geometric and industrial tests, even accounting for kinematics and distance queries, can be handled with high accuracy in reasonable computing time.

1. Introduction

The design of a workcell is a critical activity in many industrial areas, see [1]. In the automotive industry, for example, where many different car models are manufactured, the number and placement of robots for joining operations is crucial for the cycle time of the production line, see [2–4]. If one also accounts for limited space availability, position uncertainty, and energy consumption [5], then it becomes clear how important the layout design is, see Fig. 1(a).

In robotics curve following operations such as sealing, painting [6–8], cutting, welding [9], additive manufacturing [10,11], the manipulator's Tool Center Point (TCP) often needs to have controlled speed to achieve the best quality performance. This requirement constrains the robot motion to avoid singularities along the entire operation and, therefore, the initial robot base positioning is critical. Other processes requiring a careful choice of the manipulator's base include sheet layup [12], trim in shoe production [13], cleaning [14], machining [15,16] and processes in industry historically less prone to robotics, such as construction [17,18].

Similar problems arise in many other sectors, where the placement of the workpiece with respect to the machine/robot carrying out the operations on it affects feasibility, quality and throughput. These include inspection [19,20], grasping [21], pick-and-place [22,23], grit blasting [24], and, lately, medical applications [25,26].

In this article we face the general problem of placing one or several manipulators in a working cell, having as main goal to minimize the number of locations to carry out all operations. The developed algorithms can also be applied, with small adaptations, to robots mounted on wheeled platforms.

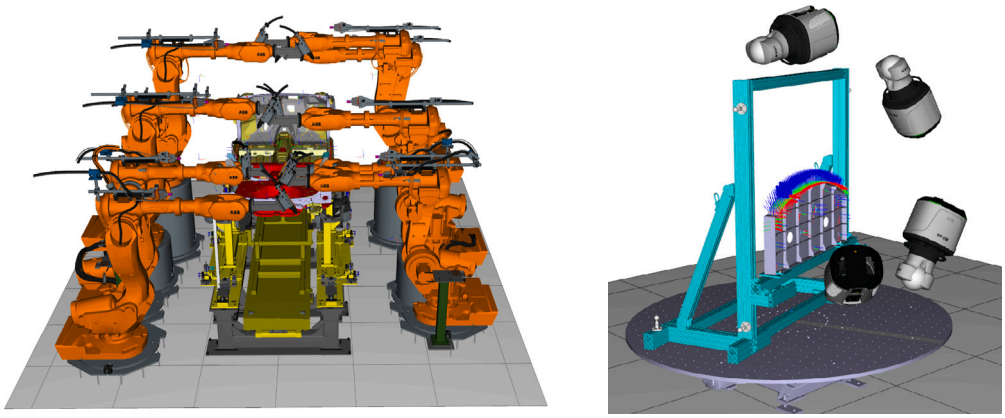
Here is a brief outlook of the article. In the next section, we will review some of the research done in the last 40 years. Section 3 outlines the contribution of this article, where the main algorithms are described. Section 4 shows the computational studies we have carried out to investigate the performance and limitations of the proposed algorithms. Section 5 summarizes the article and presents future work ideas.

2. Related work

Robot placement and feature coverage have been very active research areas in the latest years. One of the pioneering works in the

* Corresponding author.
E-mail address: domenico.spensieri@fcc.chalmers.se (D. Spensieri).

https://doi.org/10.1016/j.rcim.2024.102932
Received 1 July 2024; Received in revised form 11 October 2024; Accepted 8 December 2024
Available online 19 December 2024



(a) A stud welding station with multiple choices for robots placement. Courtesy of Volvo Cars.

(b) A laser radar inspection cell with multiple choices for sensors placement. Courtesy of Saab Aeronautics.

Fig. 1. Industrial scenarios where robot placement and inspection sensor location are relevant.

field is [27], where the authors use a manipulability index to rank different poses in the manipulator's workspace. Moreover, they also check the kinematic feasibility of motions between poses corresponding to consecutive tasks. Another early work, [28], aims to minimize cycle time for a set of tasks, by moving the manipulator location. The study is limited to a 2-dofs mechanism.

During the '90s, the area grows, following the general trend focusing on off-line programming for robot tasks. In [29] the authors improve several kinematic quality criteria by using the Box method [30], which does not require derivatives. Nelson and Donath improve their own work in [27] with [31], where a gradient-based algorithm is used to position the manipulator performing several tasks. To guide the iterations, the method transforms the manipulability index gradient w.r.t. the joint space, into a gradient w.r.t. the Cartesian space. Feddema in [32] finds the optimal location for a manipulator to minimize point-to-point motions time: the idea is to express the gradient of the motion time w.r.t. the coordinates of the robot base and use it in a steepest descent algorithm. Another impressive work is [33], where the authors minimize cycle time for a set of tasks. In addition to varying the base location for the manipulator, they also optimize the order of tasks, given a base location, by solving the Generalized Traveling Salesman Problem (GTSP), see [34]. In [35] the authors use the Nelder–Mead [36] method to optimize the motion time by varying the robot base location when the end-effector path is prescribed. A breakthrough in the field, consisting in using collision avoidance on complex CAD models was done in [37]. They use a path-planner and a path-optimization algorithm to find collision-free paths for a robot, given a base location. However, using a brute-force approach going through all possible base locations and perform path planning from scratch would require enormous amount of time to be a viable solution. Therefore, they exploit "coherence" of the configuration space, by using as starting path via-points for a new near base location, the one composed by the same robot configurations as the previous computed path. Another work computing possible base locations (in SE(2)) for complex kinematic chains is [21]. Here, a precomputed inverse reachability map is used together with a lazy collision check, even for human-like kinematic objects.

Mitsi and al. [38] use a hybrid method consisting of Genetic Algorithms (GA, [39]), quasi-Newton algorithm and constraints handling method to optimize the manipulator base location, given the end-effector poses. GAs are also used in [19,40,41]. In the latter, the manipulator base location is optimized to minimize the velocity ratio for a path-following task. The Response Surface Method is used in [42] to approximate the cycle time as a function of the robot base location and optimizing it by standard optimization algorithms. In addition to mounting a manipulator in a workcell, two similar problems are also relevant in this work, which share most of the scientific methods listed above:

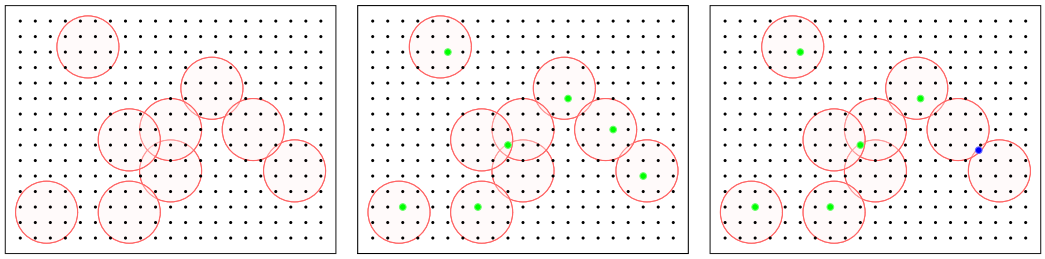
- *mobile manipulators*, where robots/manipulators are mounted on a moving platform, often wheeled;
- *features coverage*, where features such as inspection points or surfaces have to be covered by sensors for metrology applications (see Fig. 1(b)) or CAD modeling.

Mobile manipulators

A closely related problem is studied in [22], where a sequence for the base positions of a dual-arm robot is optimized. Its application comes from pick-and-place tasks for a robot positioned on a wheeled vehicle. They use Branch-and-bound (B&B, [43]) combined with Simulated Annealing (SA), [44], for the optimization part and Quadratic Programming (QP, [45]) to find collision-free inverse kinematics solutions. Pick-and-place applications for a mobile manipulator are also treated in [23]. The authors use a reachability database to explore inverse kinematic solutions, while accounting for positioning uncertainty and check for collisions in a separate step. A set covering formulation is used to reduce the number of movements. Positioning uncertainty is also treated in [17], with a related set covering model, for construction applications. Similarly, the minimum number of platform positions for a mobile robot is studied in [46], where GAs are used; collision avoidance is tackled through approximating safe areas by including equations in the general mathematical programming formulation. Another work concerning mobile manipulators is [47]. Here, the authors use a torus-based model of the robot workspace and apply their method to a manipulator, whose base is able to move on a (2D) plane. In [48], a task-oriented performance index is optimized by a large nonlinear programming model and solved through Sequential Quadratic Programming (SQP, see [45]), also including collision avoidance. This method, however, relies on the possibility to approximate geometries by regular volumes to check whether they contain a point cloud or not.

In [9] the optimal placement of a robot is studied for welding processes. The authors use a Particle Swarm Optimization (PSO, [49]) method combined with an approximation of collision avoidance constraints by an algebraic formulation.

An efficient library to optimize the base placement for manipulators is presented in [50]. They compute the reachability map and an efficient representation of it to guide the search, without, however, considering general collision avoidance.



(a) Simple instance of MGUDC with 9 disks and uniform grid.

(b) An optimal solution with 7 points in the uniform grid.

(c) An optimal solution with 6 points, adding the blue sample.

Fig. 2. MGUDC instance 2(a) solved by uniform sampling 2(b) vs. additional intersection points 2(c).

In [51] a convex programming approach is used and collision avoidance is limited to situations that can be approximated by convex functions. The same problem, in a dual perspective, is studied in [52] where the workpiece placement is optimized. A non-linear programming approach is also used in [8] for painting applications.

Feature coverage

Coverage path planning for robotics has been extensively studied, [53], related to several application areas, for example in scanning and 3D object reconstruction, such as [54–56], and [57]. In this article, the focus lies in general viewpoint optimization where the sensors can be moved by robots. Previous research identifying areas (or viewpoints) to entirely cover the workpiece to be inspected, is [58,59], where the minimum number of viewpoints is found, considering visibility, field of view, resolution and focus. Combining viewpoint optimization with control on measurement uncertainty is done in [60], where both sampling and the sequence of the resulting samples are optimized. In [61] minimizing the number of viewpoints based on targetted areas, rather than uniform or random sampling is proposed. Their work includes also constraints relative to the robot kinematics and collision avoidance. The most recent work combining mobile manipulators and features coverage is [62]. The authors propose a genetic algorithm approach to first optimize viewpoint quantity and their quality, then to assign viewpoints to stations and plan their sequence.

The aforementioned studies address feature coverage and robot positioning as distinct challenges. In contrast, we propose a unified model that simultaneously addresses both tasks while incorporating collision-free constraints at varying levels of complexity. Additionally, the proposed method introduces a novel approach to enhancing accuracy and completeness by generating problem-adapted grid samples.

3. Modeling and optimization

Modeling these problems may be done by first analyzing the Minimum Geometric Unit Disk Cover (MGUDC) problem. It involves finding the minimum number of unit disks (radius 1) required to cover a set of N points in the plane. This NP-hard problem, see [63], has applications in areas like communication networks and facility location. While many studies focus on approximation algorithms for large point sets [64,65], we are interested in exact solutions for small to medium-sized instances. We adopt a dual approach: given a set $\mathcal{F}$, with $|\mathcal{F}| = N$ of unit disks, find the smallest set of points $\mathcal{P}$ such that each disk $f \in \mathcal{F}$ contains at least one point from $\mathcal{P}$. See Fig. 2(a) for an example.

3.1. Rationale

The idea is to find all subsets $\mathcal{F}_i$ of $\mathcal{F}$ where the corresponding disks have non-empty intersection, *i.e.* $\bigcap_{f\in\mathcal{F}_i} f \neq \emptyset$. While the number of such subsets can grow exponentially, we show that, in our applications, this growth is manageable. This is the core idea from a combinatorial standpoint.

From a geometric perspective, we begin with a uniform grid of points and associate each point with a subset, say $\mathcal{F}_i$, of disks containing it. The goal is to check if there are points outside the grid covering a larger subset, say $\mathcal{F}_j \supset \mathcal{F}_i$. Fig. 2(a) shows an instance of the MGUDC problem with a uniform grid, and Fig. 2(b) shows an optimal solution with 7 points. However, some disk intersections are missed by the grid, and adding a point (in blue) in these intersections improves the solution from 7 to 6 points, as shown in Fig. 2(c).

We combine these two points of view in the algorithm below.

3.2. The base algorithm, feature coverage

Here, we present an algorithm to find all candidate points to cover a set of convex shapes in 3D, therefore even including the MGUDC problem. The main idea is to divide the search space in box-shaped cells (rectangles in the 2D case) and to keep subdividing them until:

- a point intersecting a "maximal" subset is found, *i.e.* point p representing cell q is included in all shapes intersecting cell q or
- a cell with a minimal predefined volume v_{min} is reached.

We start by defining an empty set $\mathcal{L}$ of *leaves*, representing cells, each one associated with a point p and a set $\mathcal{C} \subseteq \mathcal{F}$ of convex shapes (each one including p). The convexity assumption is needed in a step of the proposed algorithm, see Section 3.2.3.

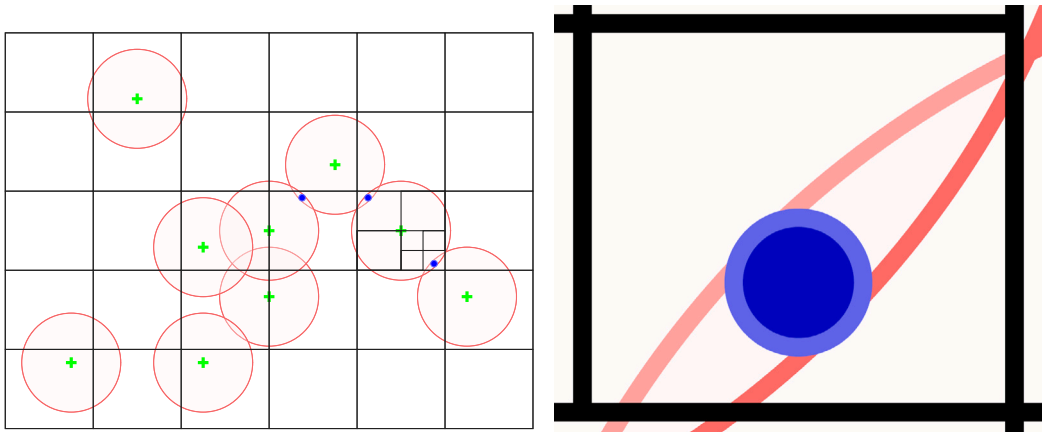
Algorithm 1 Feature Coverage

```
1: L := ∅
2: Initialize Q                                          ▷ Section 3.2.1
3: while Q ≠ ∅ do
4:     q := Q.pop()
5:     S := divide(q)                                    ▷ Section 3.2.2
6:     for s ∈ S do
7:         s.C := findCovering(s, F)                     ▷ Section 3.2.3
8:         if s.C ≠ ∅ ∧ ¬L.dominates(s) then
9:             s.p := findIntersection(s.C)              ▷ Section 3.2.4
10:            if s.p is found then                      ▷ Section 3.2.5
11:                L.insert(s)
12:            else
13:                Q.push(s)
14:            end if
15:        end if
16:    end for
17: end while
```

3.2.1. Queue initialization

A queue is used to process the cells. It is initialized at line 2, by creating a coarse grid in the search space, adapting the cell size to the problem and having a maximum of 1000 cells at the beginning. Each cell is also initialized by constructing its set $\mathcal{C}$ of *covering* shapes. If $\mathcal{C} = \emptyset$, then the cell is not pushed into the queue.

Several queue ordering strategies are possible, such as First In First Out (FIFO), as well an ordering based on the size of the cell and on the cardinality of $\mathcal{C}$. Our computational experience showed that a stack,



(a) Cell subdivision until an intersection point is found.

(b) A *leaf* cell with two intersecting shapes (in red) and a point (in blue) in their intersection area.

Fig. 3. Cell subdivision and point in the intersection of two disks.

with its Last In First Out (LIFO) strategy, performs very well in practice. However, more sophisticated strategies are possible, which prioritize the search for a good solution based on how the search space is being scanned.

3.2.2. Cell subdivision

At line 5 the current cell q is processed and subdivided into smaller ones, with a volume always larger than v_{min}. Also here, one can create an oct-tree, or divide it in 2 or several sub-cells, see Fig. 3(a). Our computational experience showed that dividing the cell along the two largest cell axes lead to very good performances. This gives a set S of sub-cells.

3.2.3. Finding shapes intersecting a cell

Each subcell $s \in S$ needs to be evaluated. In particular, its intersecting shapes have to be identified. This is an easy operation to perform when we are in the two-dimensional space with disks as shapes and rectangles as cells. However, our ambition is to be able to use the same algorithm even in three dimensions and for general convex shapes.

Thanks to the convexity property, one could formulate this problem as a mathematical programming model and solve it by general iterative methods with good performance and guarantees. An example where the convex shapes are spheres is the one below:

$$\begin{aligned} \text{minimize} \quad & g(x) && \text{(1a)}\\ \text{s.t.} \quad & \|x - c\|^2 \le r^2 && \text{(1b)}\\ & l \le x \le u && \text{(1c)}\\ & x \in \mathbb{R}^3, && \text{(1d)} \end{aligned}$$

where c is the center of the sphere and r is its radius and $l, u \in \mathbb{R}^3$ are, respectively, the lower and upper bound vectors defining the axes-aligned box. Note that the objective function g can assume several forms, depending on the goals one has. It can be a constant, to check feasibility, or one can try to find a point closest to the center.

However, we found out that this approach is too slow in many tests. Another disadvantage is that the convex shapes require an analytical formulation, often with gradient expressions as well, therefore we have implemented a faster geometrical algorithm, proposed in [66].

The basic idea is to iteratively generate candidate directions to separate the two convex sets, until a decision can be taken. We exploit the fact that we are not interested in finding a point in the intersection but only in the decision version of the problem, *i.e.* whether they do intersect or not. This is the strategy implementing the function al line 7, in the version run to generate all computational tests in this article, see Section 4.

3.2.4. Finding the shapes intersection

If $C \neq \emptyset$ for the sub-cell s, then we start verifying that there is no cell in $\mathcal{L}$ dominating s. A cell q dominates another cell s if $s.C \subseteq q.C$ and therefore can s be excluded from the search. Then, we try to find a point p in the cell s such that $p \in f, \forall f \in s.C$. In other words, we want to find an intersection of all shapes in $s.C$, see Fig. 3(b).

The same example as problem (1) with the spheres, but accounting for their intersection, can be written as:

$$\begin{aligned} \text{minimize} \quad & g(x) && && \text{(2a)}\\ \text{s.t.} \quad & \|x - c_f\|^2 \le r_f^2 && f \in F && \text{(2b)}\\ & l \le x \le u && && \text{(2c)}\\ & x \in \mathbb{R}^3 && && \text{(2d)} \end{aligned}$$

where F is the set of indices $\{1, 2, \dots, |\mathcal{F}|\}$ of all shapes to be covered (or robot tasks to be performed), and each sphere has center c_f and radius r_f.

This approach has the same drawbacks as problem (1), *i.e.* requires analytical formulation for convex shapes and is slow. Therefore, in practice, it is more efficient to sample the cell and check if there are points contained in all shapes. If this is not the case, the cell needs further subdivision and is added to the queue (see line 13). This is the strategy implemented the function at line 9, in the version run to generate all computational tests in this article, see Section 4.

Note that the difference between the mathematical programming approach and the approximation scheme is that in the first case computational time is spent to solve the mathematical problem, whereas the second case copes with approximation by refining the cell.

3.2.5. Update

If a common point to all shapes is found, then its corresponding cell can be added to the set $\mathcal{L}$ of leaves. Also in this operation, we test whether the newly inserted cell dominates some of the cells already present. In this case, they are removed and replaced by the new cell.

So far, we have assumed that there is no impairment at all when placing points in space to cover a feature. This might be true for some applications where one neglects geometrical information. However, we are here interested in feature coverage by a laser radar, which needs to have free sight to the workpiece surface. Moreover, we are also interested in robot applications where reachability and distance requirements to the environment are critical.

3.3. The enhanced algorithm

Accounting for these two requirements at the same time can be done by adding the corresponding test, see line 16 in Algorithm 2, when a leaf is found, *i.e.* when a point included in all intersecting features is found. However, Algorithm 1 does not catch geometrical information regarding the extra requirements, therefore one might end up generating cells of minimum resolution, despite fulfilling the *leaf* criterion. In this case, then, one tries to create candidates by retrieving more information through the additional tests required, see line 7 in Algorithm 2, which we name *Enhanced Feature Coverage*.

Algorithm 2 Enhanced Feature Coverage: distance queries and inverse kinematics integration

```
1: L := ∅
2: Initialize Q
3: while Q ≠ ∅ do
4:     q := Q.pop()
5:     S := divide(q)
6:     if S = ∅ then
7:         q.C := findAllVisible(q.p, F)
8:         if q.C ≠ ∅ ∧ ¬L.dominates(q) then
9:             L.insert(q)
10:        end if
11:    end if
12:    for s ∈ S do
13:        s.C := findCovering(s, F)
14:        if s.C ≠ ∅ ∧ ¬L.dominates(s) then
15:            s.p := findIntersection(s.C)
16:            A := findAllVisible(s.p, F)
17:            if s.p is found ∧ s.C ⊆ A then
18:                L.insert(s)
19:            else
20:                Q.push(s)
21:            end if
22:        end if
23:    end for
24: end while
```

The *findAllVisible* routine needs to be customized for each application. In the case of feature coverage by a laser sensor, one needs to check that the sensor geometry itself has no collision with the environment and that the laser beam is collision-free as well. For robot applications one needs to solve inverse kinematics and compute distance w.r.t. the rest of the scene to guarantee the process clearance requirements (see Fig. 4).

3.4. Set covering

When the queue is empty, it means there are no more cells left to process. At this point, we have found all the relevant candidates needed to cover all the features. Specifically, we have identified all the shapes, denoted as the subset $\mathcal{F}_i$, that intersect with each cell, and we have verified whether all these shapes have at least one intersection point within the cell. This ensures that no other cell (with a volume greater than our threshold) exists that covers a subset of the shapes in $\mathcal{F}$ that has not already been identified.

Since we are interested in finding the minimum number n^* of locations, one can formulate a set covering problem. By defining a set L of *leaf* indices $\{1, 2, \dots, |\mathcal{L}|\}$, and a function $C(f)$ retrieving all leaf indices $i \in L$ covering feature $f \in F$, one can write it as

$$\text{minimize} \quad \sum_{i \in L} x_i \tag{3a}$$

$$\text{s.t.} \quad \sum_{i \in C(f)} x_i \geq 1 \qquad f \in F \tag{3b}$$

$$x_i \in \{0, 1\} \qquad i \in L \tag{3c}$$

The optimization step can also be used to terminate processing the cells from the queue if one can prove that no other cell currently in the queue will decrease the optimum value found so far. To do that, we can keep track of an upper bound $\overline{n}$ and a lower bound $\underline{n}$ at each iteration ($\underline{n} \leq n^* \leq \overline{n}$). The upper bound can be computed as in Eq. (3) with the current set of leaves. The lower bound is obtained in a similar way, by defining a set of indices $Q = \{|\mathcal{L}| + 1, |\mathcal{L}| + 2, \dots, |\mathcal{L}| + |Q|\}$ for the cells in the queue and a function $G(f)$ retrieving all indices $i \in Q$ corresponding to cells in Q covering feature $f \in F$:

$$\text{minimize} \quad \sum_{i \in L \cup Q} x_i \tag{4a}$$

$$\text{s.t.} \quad \sum_{i \in C(f) \cup G(f)} x_i \geq 1 \qquad f \in F \tag{4b}$$

$$x_i \in \{0, 1\} \qquad i \in L \cup Q \tag{4c}$$

The strategy adopted is to carry out the update of $\overline{n}$ and $\underline{n}$ once in a while and to stop the algorithm if $\overline{n} = \underline{n}$.

Keeping track of these bounds can also help us to obtain high-quality solutions fast. Indeed, one can design the queue Q in a way to favor cells with a potential to decrease the current optimum n. So, if a time limit is reached, then a good solution can still be retrieved. Moreover, if the current optimum has not been improved during many iterations, it is possible to switch to a queue pattern where cells with a potential of increasing the lower bound are prioritized. We refer to the algorithm where the set covering is solved along the iterations as Algorithm 3 or *Lazy Enhanced Feature Coverage*, since we try to terminate as soon as possible. In practice, this strategy allows to obtain high-quality solutions with a small gap to the optimum for instances where distance queries become computationally heavy.

Note that, we strive to keep the queue size low, therefore the problem above is not very difficult to solve by general Integer Linear Programming solvers, such as HiGHS, see [67]. If this is not the case, one can relax the variables to be $x_i \in [0, 1]$. This relaxation may produce less tight lower bounds.

3.5. Implementation notes and speed-up

Reachability analysis for the robots, their inverse kinematics solutions and distance computations are done by using built-in functionalities in the robotics simulation software Industrial Path Solutions, see [68].

All the parts of the algorithm have been implemented in C++20, with extensive use of the C++ Standard Library and built with Microsoft Visual Studio.

The first improvement to the algorithm presented is to implement the queue to support parallel threads running on it. This allows us to process each cell from the queue in its own thread. The choice about the maximum number of threads is machine dependent.

Another optimization that can be done is to carry out as first distance computation for the children cells, the ones which resulted in a collision for the parent cell. In case there is still collision, one does not need to carry out tests for the remaining shapes, since the cells need to be split anyway.

4. Computational results

In this Section we describe and report and results of the computational study we have performed to test the algorithm in different scenarios. We have run:

- Algorithm 1, *Feature Coverage*, for instances without collision, see Section 4.1;

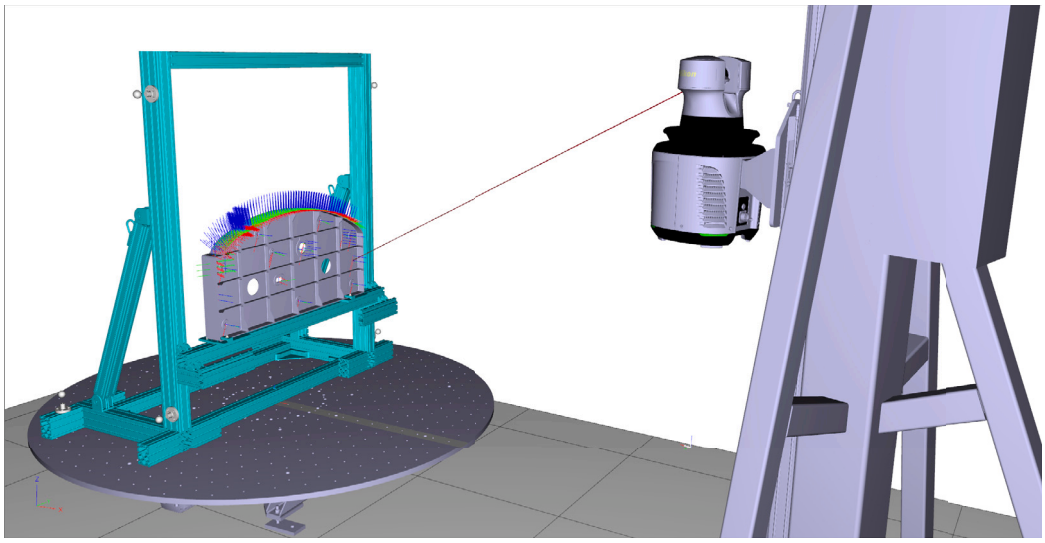
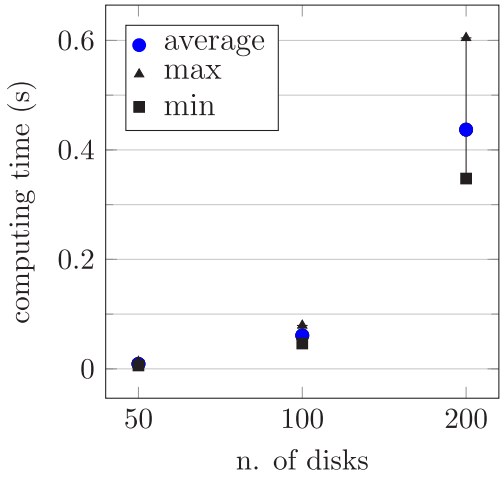
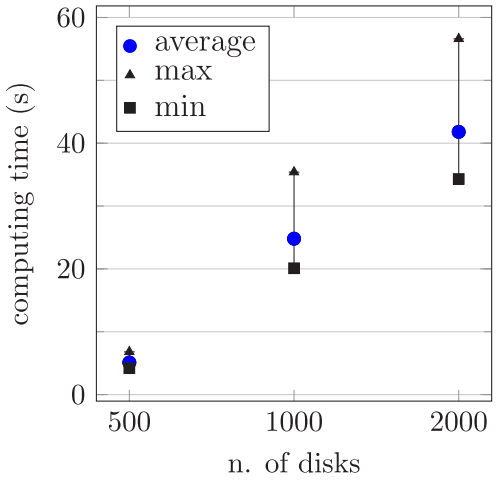


Fig. 4. The laser sensor needs to have free sight to the workpiece, *i.e.* the laser beam should be collision-free w.r.t. the environment.



(a) Times to produce candidate points, for problems with 50, 100, and 200 randomly-positioned unit disks.



(b) Times to produce candidate points, for problems with 500, 1000, and 2000 randomly-positioned unit disks.

Fig. 5. Candidates generation times (s) for MGUD instances of varying size, with randomly-positioned unit disks.

- Algorithm 2, *Enhanced Feature Coverage*, for instances where the collision queries are very fast, see Section 4.2;
- Algorithm 3, *Lazy Enhanced Feature Coverage*, when distance queries dominate the computation time, see Section 4.3.

We have highlighted in Section 2 that many similar approaches have been proposed, each with its own twist: minimize energy, maximize manipulability, process quality, etc. A common strategy to many works is to discretize the search space, see [3,7,14,22], therefore we have compared our proposed algorithms with a uniform discretization of the search space, followed by the solution of a set covering problem.

All the experiments have been carried out on a desktop machine running Windows 11, with an AMD Ryzen 9 3900X Twelve-Core Processor 3.7 GHz and 32 GB of installed RAM.

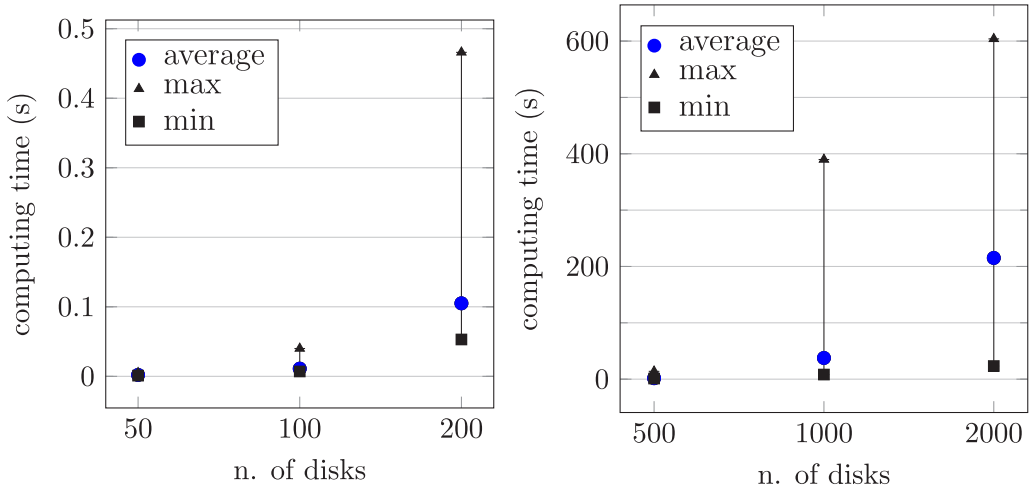
4.1. Geometrical instances with no collisions

4.1.1. 2D: MGUDC

The first scenario is the MGUDC problem. Instances have been generated by creating a 10 × 10 square with randomly positioned unit disks. The v_{min} parameter, minimum cell area in these cases, has been fixed to 1e−5 and 100 random instances have been generated for 50, 100, 200, 500, 1000, 2000 disks. The minimum, maximum and average solving times for Algorithm 1 are illustrated in Fig. 5, whereas HiGHS performance on the resulting set covering problems is depicted in Fig. 6. Note that all these problem sizes can easily be managed and the bottleneck, in terms of computing times, is represented by finding the optimal solution to the set covering problem for very large instances, with about a factor 10 difference w.r.t. the generation of candidates.

4.1.2. 3D: Cones

In this scenario, we have created 427 inspection points needed to be measured by a laser scanner with a given maximum angle between the beam and the surface normal. This gives raise to 427 cones as shapes, see Fig. 7, in this paper usually denoted as $\mathcal{F}$. We compare the performance of Algorithm 1, *Feature Coverage* against a brute-force approach where the workspace is uniformly sampled, with the same resolution as the minimum evaluated cell size in our algorithm. In Fig. 8(a) it is possible to see the minimum number of locations needed to cover all



(a) Times to solve the resulting ILP, for the instances used in Figure 5a.

(b) Times to solve the resulting ILP, for the instances used in Figure 5b.

Fig. 6. Solving times for resulting ILP, for MGUD instances of varying size, with randomly-positioned unit disks.

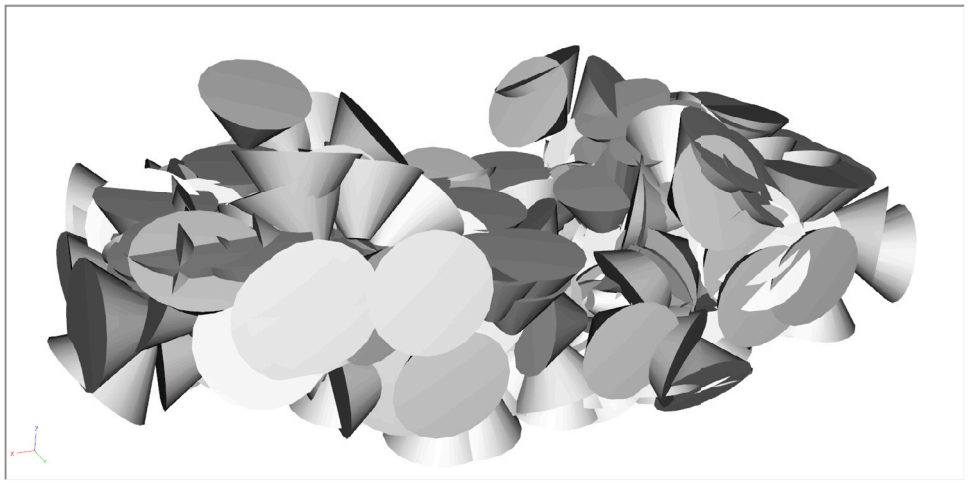
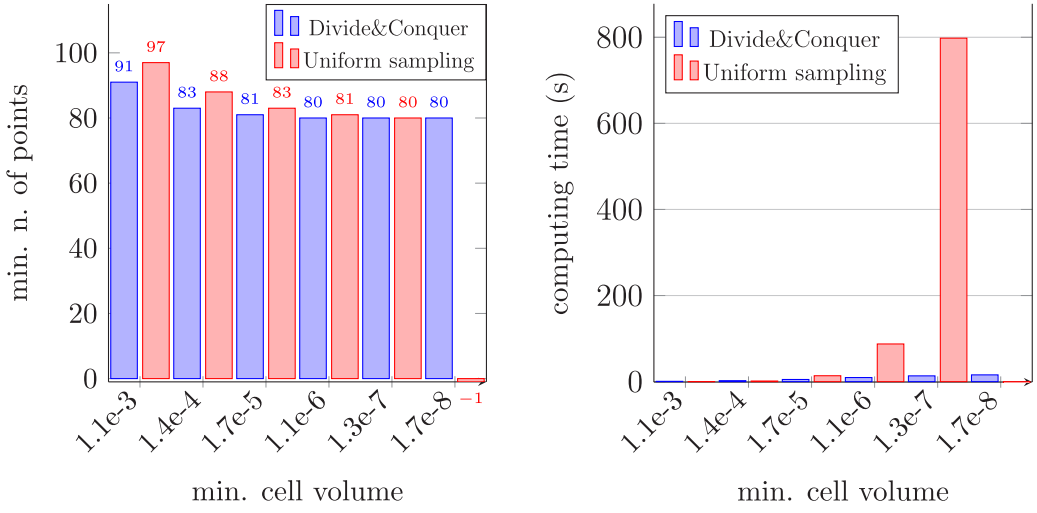


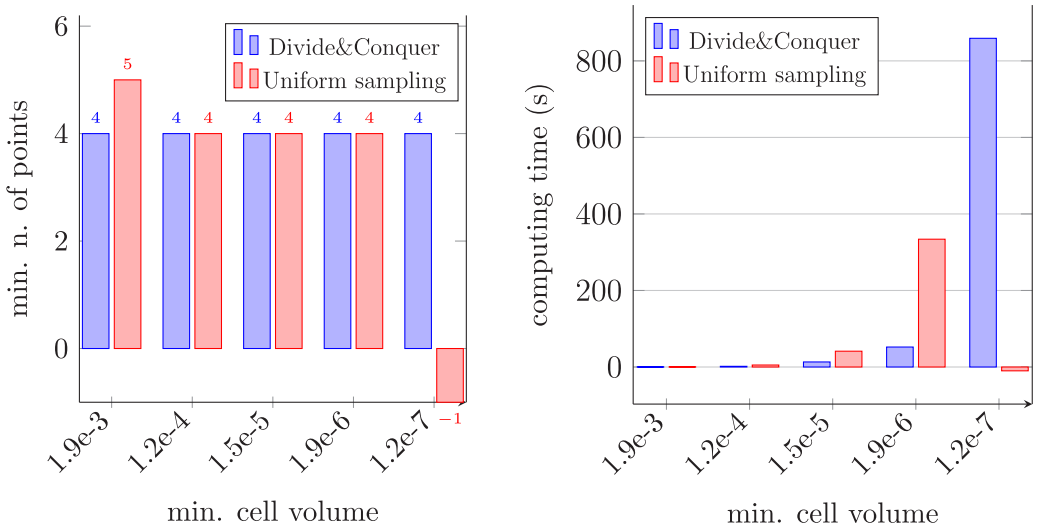
Fig. 7. 427 cone features placed on a car frame (hidden).



(a) Min. n. of points found to cover all cones. Negative number represents no optimum found.

(b) Computing times to produce candidate points. Negative number represents no optimum found.

Fig. 8. Optima and candidates generation times (s) for the case with 427 cones and no visibility test. Negative number represents out of memory.



(a) Min. n. of points found to cover all cones. Negative number represents no optimum found.

(b) Computing times to produce candidate points. Negative number represents no optimum found.

Fig. 9. Optima and candidates generation times (s) for the sensor placement at Saab Aeronautics. Negative number represents out of memory.

cones when varying resolution: due to the more sophisticated sampling strategy the optimum found is always better or equal to the one for the uniform sampling approach. Moreover, Fig. 8(b) shows running times become more and more favorable when increasing the resolution. The minimal cell volume results in an excessive number of samples, leading to memory overflow.

4.2. Industrial case with ray-casting

4.2.1. What is the minimum number of sensor locations to cover all tasks?

This case consists of 413 inspection points distributed on an artefact at Saab Aeronautics in Sweden to develop new metrology strategies, see Fig. 1(b) which also shows the 4 optimal sensor locations found. Even in this case, the feature shapes input to the algorithms are cones, since the laser beam has measurement restrictions on the angle built with the product surface. Algorithm 2 or *Enhanced Feature Coverage* has been chosen to handle the extra-requirement for the laser beam, and no lazy strategy is required due to the fast implementation for the collision test and rare laser occlusion occurrence. Fig. 9(b) shows the computing times to run it comparing it to a uniform sampling strategy, at different resolutions. It is evident that our algorithm is able to evaluate less cells and the effect is highlighted when the minimum cell allowed becomes smaller. Memory was not enough for uniform sampling and $v_{min} = 1.2e-7$. The minimum numbers of locations found by the two algorithms are shown in Fig. 9(a).

4.3. Industrial case with distance queries

In the case of typical industrial manipulators with six degrees of freedom, such as the robots analyzed in these industrial scenarios, their workspace can be conservatively overestimated by a sphere. Consequently, following the dual approach utilized throughout this paper, the convex sets $\mathcal{F}$ are represented as spheres positioned at the locations where the robot faceplate must be situated to perform the operations, specifically cleaning curve following and stud welding in this context. Note, here, that the feature shape depends on the robot workspace and the operation to be performed, thus it is not an exclusive property of the process itself as in the previous cases. Moreover, the feature positioning depends on the tool used since it defines the position and orientation of the robot faceplate.

4.3.1. Where should the AGV/positioner stop to perform all cleaning operations?

The proposed algorithm can also be used to determine the positions of an AGV with a manipulator, which has also the possibility to be raised vertically to increase reachability, see Fig. 10(a). The overestimated robot workspace is depicted in Fig. 10(b) and centered at all required 35 cleaning locations in Fig. 10(c).

Now, the collision queries start to become relevant, so we adopt the Algorithm 3, *Lazy Enhanced Feature Coverage*, where the set covering is solved once in a while during the generation of the candidates.

The optimal locations found are illustrated in Fig. 10(d) and the comparison between the number of locations found by the proposed algorithm and he uniform sampling is depicted in Fig. 11(a). Note that this kind of application requires few locations with many possibilities, therefore there is no differences between the optima found.

On the other hand, Fig. 11(b) shows the computing times to run it in comparison to a uniform sampling strategy, at different resolutions. This scenario shows the same trend as in Section 4.2 in favor of our adaptive grid algorithm.

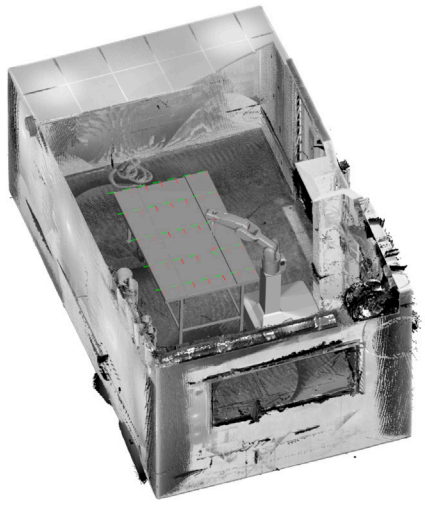
4.3.2. How many robots are needed to cover/carry out all stud welding tasks?

We have also studied the case where one needs to determine the number of robots required to carry out 77 welding tasks on a body-in-white, see Fig. 12(a). This scenario is challenging due to computationally heavy distance queries because of the cluttered environment. The overestimated robot workspace is depicted in Fig. 12(b) and centered at all stud locations in Fig. 12(c). The solution found as an optimal one is illustrated in Fig. 12(d).

The optima found between the two algorithms are the same, see Fig. 13(a), and in Fig. 13(b) it is possible to see how the gain is not as evident as for the previous scenarios. This is due to the fact that the distance function is not caught or tried to be reconstructed along the proposed algorithms. Moreover, the coexistence of multiple robots requires more work to handle mutual collisions.

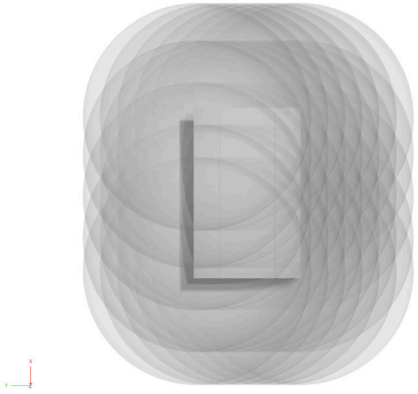
5. Discussion and future work

We have provided a divide&conquer algorithm to minimize the number of sensor/camera locations to inspect a set of features on a workpiece. The algorithm has been generalized to tackle positioning of

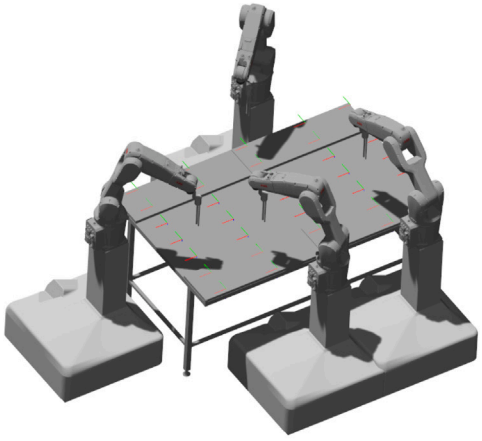


(a) Cleaning manipulator on AGV.

(b) Workspace for the ABB robot used.

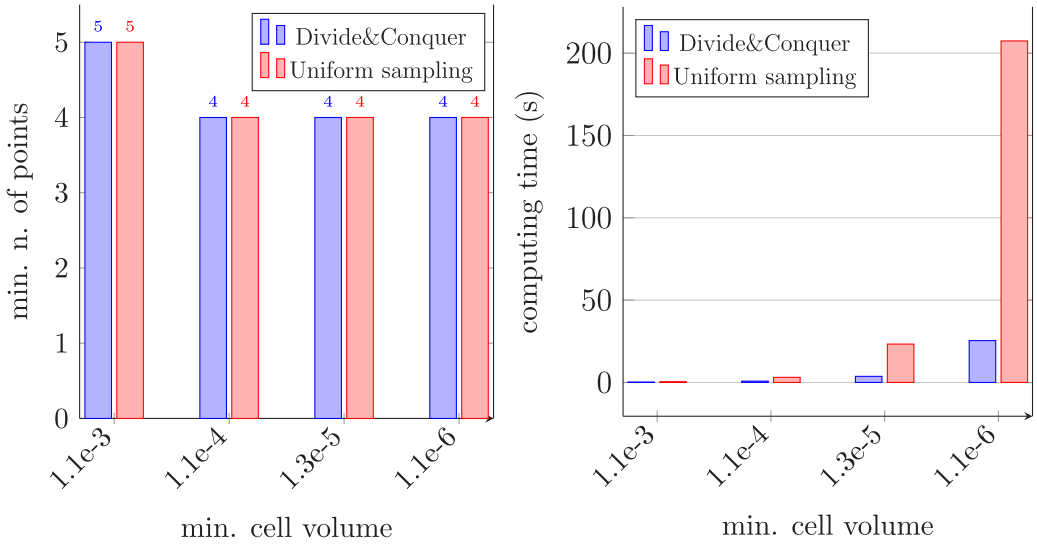


(c) Robot workspace centred at all 35 cleaning task locations.



(d) An optimal solution found, consisting of 4 locations, with relative height.

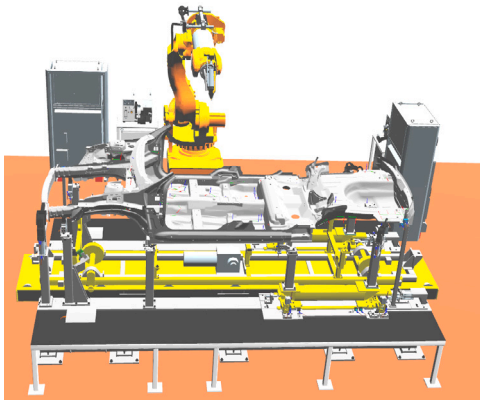
Fig. 10. Illustration of different aspects of the cleaning station application.



(a) Min. n. of locations found to cover all cleaning tasks.

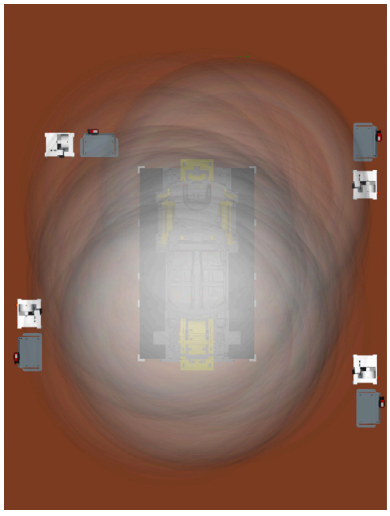
(b) Candidates generation times (s) for the manipulator on AGV.

Fig. 11. Optima and candidates generation times (s) for the cleaning station application.

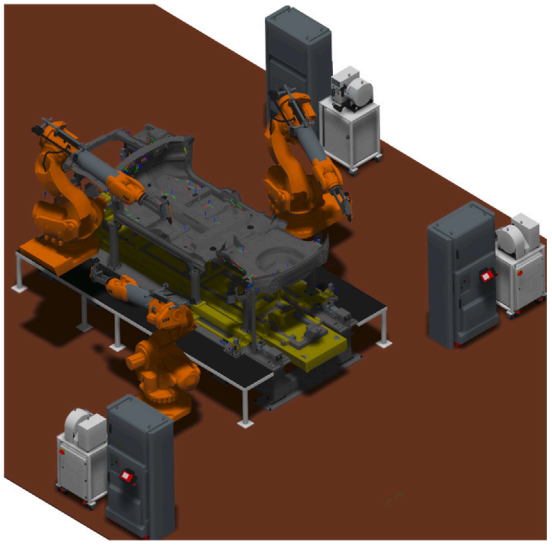


(a) Stud welding station with an industrial robot.

(b) ABB robot with stud welding tool and its approximated workspace.

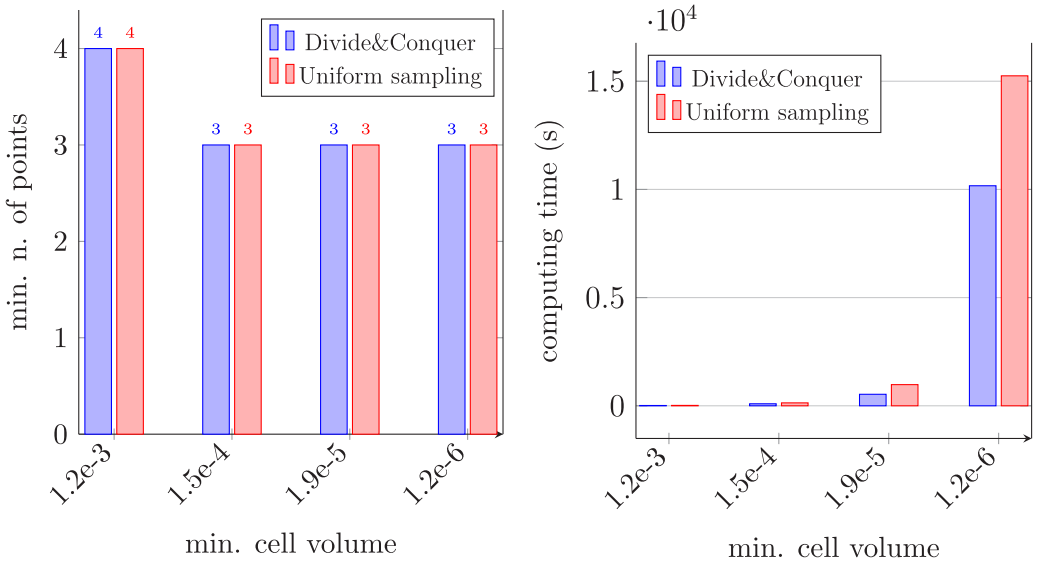


(c) Robot workspace centred at all 77 stud locations.



(d) An optimal solution found, consisting of 3 robots and their positions.

Fig. 12. Illustration of different aspects of the stud welding station application.



(a) Min. n. of robot found to cover all 77 stud tasks.

(b) Candidates generation times (s).

Fig. 13. Stud welding application in the automotive industry with 77 welding tasks on body-in-white.

mobile manipulators and deciding the number of robots needed to carry out a set of operations in a station. Computational experiments have proven the algorithm to be efficient in handling cases where collisions are not very relevant in the application, whereas for high-cluttered scenarios we have introduced an optimization loop to save computing time. Several speed-up strategies have been presented, however the generalized problem still presents some challenges.

The critical performance criterion is cycle time, which is also influenced by how the set of operations is distributed among the robots. Fixing the robot locations and optimizing robot loads, task sequences and robot motions can already be efficiently solved today, see [69–71]. However, simultaneously optimizing robot positions and load balancing is challenging and is a topic for future research.

Another important mechanism is the proof of optimality run throughout the algorithm to stop computations. More sophisticated approaches may be devised, which will be investigated in the near future.

In summary, the current algorithm solves three closely related problems and can already be integrated in CAx software to solve industrial cases from design to manufacturing/inspection planning, contributing to cut costs, reduce cycle time and improve measurement robustness.

CRediT authorship contribution statement

Domenico Spensieri: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Edvin Åblad:** Writing – review & editing, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Raad Salman:** Writing – review & editing, Validation, Software, Methodology, Data curation, Conceptualization. **Johan S. Carlson:** Writing – review & editing, Funding acquisition, Conceptualization.

Declaration about the use of AI

During the preparation of this work the authors used ChatGPT in order to improve readability and correct grammatical errors. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Declaration of competing interest

Certain commercial systems and software identified in this paper do not imply recommendation or endorsement by FCC or by the authors, nor does it imply that the products identified are necessarily the best available for the purpose.

This holds even if the authors might have employment and/or ownership relationship with the software organization.

Acknowledgments

We would like to thank the colleagues at the Fraunhofer-Chalmers and the project partners at Saab Aeronautics for useful discussions and help in the realization of this work. This work benefited from funding from the Swedish Innovation Agency Vinnova, as part of its Digi-Q research project, and from the Swedish Research Council for Sustainable Development Formas as part of its project “Automated open cleaning for the food industry”. It is also part of the Sustainable Production Initiative (SPI) and the Production Area of Advance at Chalmers University of Technology, Sweden. We thank also the anonymous reviewers for their comments, which contributed to improve the article.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.rcim.2024.102932.

Data availability

No data was used for the research described in the article.

References

[1] P. Franceschi, S. Mutti, N. Pedrocchi, Optimal design of robotic work-cell through hierarchical manipulability maximization, Robot. Comput.-Integr. Manuf. 78 (2022) 102401.

[2] D. Spensieri, J.S. Carlson, R. Bohlin, J. Kressin, J. Shi, Optimal robot placement for tasks execution, Procedia CIRP 44 (2016) 395–400, http://dx.doi.org/10.1016/j.procir.2016.02.105, URL https://www.sciencedirect.com/science/article/pii/S2212827116003875, 6th CIRP Conference on Assembly Technologies and Systems (CATS).

[3] M. Hassan, D. Liu, G. Paul, Modeling and stochastic optimization of complete coverage under uncertainties in multi-robot base placements, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2016, pp. 2978–2984.

[4] D. Spensieri, E. Åblad, R. Bohlin, J.S. Carlson, R. Söderberg, Modeling and optimization of implementation aspects in industrial robot coordination, Robot. Comput.-Integr. Manuf. 69 (2021) 102097, http://dx.doi.org/10.1016/j.rcim.2020.102097, URL https://www.sciencedirect.com/science/article/pii/S0736584520303070.

[5] R. Ur-Rehman, S. Caro, D. Chablat, P. Wenger, Path placement optimization of manipulators based on energy consumption: application to the orthoglide 3-axis, Trans. Canadian Soc. Mech. Eng. 33 (3) (2009) 523–541.

[6] S. Ren, X. Yang, G. Wang, Z. Liu, Q. Yu, K. Chen, Study on the stop position of the mobile manipulator for painting on big parts, 2015, http://dx.doi.org/10.1115/IMECE2015-50907.

[7] S. Ren, Y. Xie, X. Yang, J. Xu, G. Wang, K. Chen, A method for optimizing the base position of mobile painting manipulators, IEEE Trans. Autom. Sci. Eng. 14 (1) (2017) 370–375, http://dx.doi.org/10.1109/TASE.2016.2612694.

[8] Q. Yu, G. Wang, X. Hua, S. Zhang, L. Song, J. Zhang, K. Chen, Base position optimization for mobile painting robot manipulators with multiple constraints, Robot. Comput.-Integr. Manuf. 54 (2018) 56–64, http://dx.doi.org/10.1016/j.rcim.2018.05.007, URL https://www.sciencedirect.com/science/article/pii/S0736584517301539.

[9] N.C.N. Doan, W. Lin, Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6R articulated robots, Robot. Comput.-Integr. Manuf. 48 (2017) 233–242.

[10] P.M. Bhatt, A. Kulkarni, R.K. Malhan, S.K. Gupta, Optimizing part placement for improving accuracy of robot-based additive manufacturing, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 859–865.

[11] P.M. Bhatt, A. Nycz, S.K. Gupta, Optimizing multi-robot placements for wire arc additive manufacturing, in: 2022 International Conference on Robotics and Automation, ICRA, IEEE, 2022, pp. 7942–7948.

[12] R. Malhan, A. Kabir, B. Shah, T. Centea, S. Gupta, Determining feasible robot placements in robotic cells for composite prepreg sheet layup, 2019, http://dx.doi.org/10.1115/MSEC2019-3003.

[13] T. Weingartshofer, C. Hartl-Nesic, A. Kugi, Optimal TCP and robot base placement for a set of complex continuous paths, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 9659–9665.

[14] H. Zhang, K. Mi, Z. Zhang, Base placement optimization for coverage mobile manipulation tasks, 2023, arXiv preprint arXiv:2304.08246.

[15] S. Caro, S. Garnier, B. Furet, A. Klimchik, A. Pashkevich, Workpiece placement optimization for machining operations with industrial robots, in: 2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, IEEE, 2014, pp. 1716–1721.

[16] C. Dumas, S. Caro, S. Garnier, B. Furet, Workpiece placement optimization of six-revolute industrial serial robots for machining operations, in: Applied Fluid Mechanics; Electromechanical Systems and Mechatronics; Advanced Energy Systems; Thermal Engineering; Human Factors and Cognitive Engineering, in: Engineering Systems Design and Analysis, vol. 2, 2012, pp. 419–428, http://dx.doi.org/10.1115/ESDA2012-82559.

[17] D.-J. Xie, L.-D. Zeng, Z. Xu, S. Guo, G.-H. Cui, T. Song, Base position planning of mobile manipulators for assembly tasks in construction environments, Adv. Manuf. 11 (1) (2023) 93–110.

[18] A. Gienger, C. Stein, A.P.R. Lauer, O. Sawodny, C. Tarín, Data-based reachability analysis and optimized robot positioning for co-design of construction processes, in: 2024 IEEE/SICE International Symposium on System Integration, SII, 2024, pp. 1247–1252, http://dx.doi.org/10.1109/SII58957.2024.10417196.

[19] X.-Q. Lin, J.-Z. Yang, Y. Yue, B. Zhang, A base position planning strategy for a mobile inspection robot, Yuhang Xuebao/J. Astronaut. 39 (2018) 1031–1038, http://dx.doi.org/10.3873/j.issn.1000-1328.2018.09.011.

[20] H. Tugal, K. Cetin, Y. Petillot, M. Dunnigan, M.S. Erden, Contact-based object inspection with mobile manipulators at near-optimal base locations, Robot. Auton. Syst. 161 (2023) 104345.

[21] N. Vahrenkamp, T. Asfour, R. Dillmann, Robot placement based on reachability inversion, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 1970–1975.

[22] K. Harada, T. Tsuji, K. Kikuchi, K. Nagata, H. Onda, Y. Kawai, Base position planning for dual-arm mobile manipulators performing a sequence of pick-and-place tasks, in: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 2015, pp. 194–201, http://dx.doi.org/10.1109/HUMANOIDS.2015.7363551.

[23] J. Xu, K. Harada, W. Wan, T. Ueshiba, Y. Domae, Planning an efficient and robust base sequence for a mobile manipulator performing multiple pick-and-place tasks, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 11018–11024.

[24] M. Hassan, D. Liu, G. Paul, S. Huang, An approach to base placement for effective collaboration of multiple autonomous industrial robots, in: 2015 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2015, pp. 3286–3291.

[25] A. Trabelsi, J. Sandoval, A. Mlika, S. Lahouar, S. Zeghloul, J. Cau, M.A. Laribi, Optimal multi-robot placement based on capability map for medical applications, in: International Conference on Robotics in Alpe-Adria Danube Region, Springer, 2022, pp. 333–342.

[26] A. Trabelsi, J. Sandoval, A. Mlika, S. Lahouar, S. Zeghloul, M.A. Laribi, Robot base placement and tool mounting optimization based on capability map for robot-assistant camera holder, Robotica (2024) 1–22, http://dx.doi.org/10.1017/S0263574724000870.

[27] B. Nelson, K. Pederson, M. Donath, Locating assembly tasks in a manipulator's workspace, in: Proceedings. 1987 IEEE International Conference on Robotics and Automation, Vol. 4, 1987, pp. 1367–1372, http://dx.doi.org/10.1109/ROBOT.1987.1087788.

[28] B. Fardanesh, J. Rastegar, Minimum cycle time location of a task in the workspace of a robot arm, in: Proceedings of the 27th IEEE Conference on Decision and Control, IEEE, 1988, pp. 2280–2283.

[29] G. Pamanes, S. Zeghloul, Optimal placement of robotic manipulators using multiple kinematic criteria, in: Proceedings. 1991 IEEE International Conference on Robotics and Automation, IEEE Computer Society, 1991, pp. 933–934.

[30] M.J. Box, A new method of constrained optimization and a comparison with other methods, Comput. J. 8 (1) (1965) 42–52, http://dx.doi.org/10.1093/comjnl/8.1.42.

[31] B. Nelson, M. Donath, Optimizing the location of assembly tasks in a manipulator's workspace, J. Robot. Syst. 7 (6) (1990) 791–811.

[32] J.T. Feddema, Kinematically optimal robot placement for minimum time coordinated motion, in: Proceedings of IEEE International Conference on Robotics and Automation, Vol. 4, IEEE, 1996, pp. 3395–3400.

[33] Y.K. Hwang, P.A. Watterberg, Optimizing robot placement for visit-point tasks, in: Proc. AAAI Workshop on Artificial Intelligence for Manufacturing, 1996, pp. 81–86.

[34] G. Gutin, D. Karapetyan, Generalized traveling salesman problem reduction algorithms, 2009, arXiv:0804.0735.

[35] M.B. Trabia, M. Kathari, Placement of a manipulator for minimum cycle time, J. Robot. Syst. 16 (8) (1999) 419–431.

[36] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (1965) 308–313, URL https://api.semanticscholar.org/CorpusID:2208295.

[37] D. Hsu, J.-C. Latcombe, S. Sorkin, Placing a robot manipulator amid obstacles for optimized execution, in: Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99)(Cat. No. 99TH8470), IEEE, 1999, pp. 280–285.

[38] S. Mitsi, K.-D. Bouzakis, D. Sagris, G. Mansour, Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm, Robot. Comput.-Integr. Manuf. 24 (1) (2008) 50–59, http://dx.doi.org/10.1016/j.rcim.2006.08.003, URL https://www.sciencedirect.com/science/article/pii/S0736584506001025.

[39] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, USA, 1998.

[40] N. Aspragathos, S. Foussias, Optimal location of a robot path when considering velocity performance, Robotica 20 (2002) 139–147, http://dx.doi.org/10.1017/S0263574701003708.

[41] A. Nektarios, N.A. Aspragathos, Optimal location of a general position and orientation end-effector's path relative to manipulator's base, considering velocity performance, Robot. Comput.-Integr. Manuf. 26 (2) (2010) 162–173.

[42] B. Kamrani, V. Berbyuk, D. Wäppling, U. Stickelmann, X. Feng, Optimal robot placement using response surface method., Int. J. Adv. Manuf. Technol. 44 (2009).

[43] J. Clausen, Branch and bound algorithms-principles and examples, 2003, URL https://api.semanticscholar.org/CorpusID:16580792.

[44] P.M. Pardalos, T.D. Mavridou, Simulated annealing, in: Encyclopedia of Optimization, Springer US, Boston, MA, 2009, pp. 3591–3593, http://dx.doi.org/10.1007/978-0-387-74759-0_617.

[45] J. Nocedal, S.J. Wright, Numerical Optimization, second ed., Springer, New York, NY, USA, 2006.

[46] S. Vafadar, A. Olabi, M.S. Panahi, Optimal motion planning of mobile manipulators with minimum number of platform movements, in: 2018 IEEE International Conference on Industrial Technology, ICIT, 2018, pp. 262–267, http://dx.doi.org/10.1109/ICIT.2018.8352187.

[47] M. Forstenhäusler, T. Wetner, K. Dietmayer, Optimized mobile robot positioning for better utilization of the workspace of an attached manipulator, in: 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM, IEEE, 2020, pp. 2074–2079.

[48] Q. Fan, Z. Gong, B. Tao, Y. Gao, Z. Yin, H. Ding, Base position optimization of mobile manipulators for machining large complex components, Robot. Comput.-Integr. Manuf. 70 (2021) 102138.

[49] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948 vol.4, http://dx.doi.org/10.1109/ICNN.1995.88968.

[50] A. Makhal, A.K. Goins, Reuleaux: Robot base placement by reachability analysis, in: 2018 Second IEEE International Conference on Robotic Computing, IRC, IEEE, 2018, pp. 137–142.

[51] S.-W. Son, D.-S. Kwon, A convex programming approach to the base placement of a 6-DOF articulated robot with a spherical wrist, Int. J. Adv. Manuf. Technol. 102 (2019) 3135–3150.

[52] R.K. Malhan, A.M. Kabir, B. Shah, S.K. Gupta, Identifying feasible workpiece placement with respect to redundant manipulator for complex manufacturing tasks, in: 2019 International Conference on Robotics and Automation, ICRA, 2019, pp. 5585–5591, http://dx.doi.org/10.1109/ICRA.2019.8794353.

[53] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robot. Auton. Syst. 61 (2013) 1258–1276, http://dx.doi.org/10.1016/j.robot.2013.09.004.

[54] X. Fan, L. Zhang, B. Brown, S. Rusinkiewicz, Automated view and path planning for scalable multi-object 3D scanning, ACM Trans. Graph. 35 (6) (2016) 1–13.

[55] W. Jing, D. Deng, Y. Wu, K. Shimada, Multi-uav coverage path planning for the inspection of large and complex structures, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 1480–1486.

[56] M. Lauri, J. Pajarinen, J. Peters, S. Frintrop, Multi-sensor next-best-view planning as matroid-constrained submodular maximization, IEEE Robot. Autom. Lett. 5 (4) (2020) 5323–5330.

[57] S. Pan, H. Hu, H. Wei, Scvp: Learning one-shot view planning via set covering for unknown object reconstruction, IEEE Robot. Autom. Lett. 7 (2) (2022) 1463–1470.

[58] R. Raffaeli, M. Mengoni, M. Germani, F. Mandorli, Off-line view planning for the inspection of mechanical parts, Int. J. Interact. Design Manuf. (IJIDeM) 7 (2013) 1–12.

[59] W. Sheng, N. Xi, J. Tan, M. Song, Y. Chen, Viewpoint reduction in vision sensor planning for dimensional inspection, in: IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003, Vol. 1, IEEE, 2003, pp. 249–254.

[60] Y. Liu, W. Zhao, H. Liu, Y. Wang, X. Yue, Coverage path planning for robotic quality inspection with control on measurement uncertainty, IEEE/ASME Trans. Mechatronics 27 (5) (2022) 3482–3493.

[61] E. Glorieux, P. Franciosa, Coverage path planning with targetted viewpoint sampling for robotic free-form surface inspection, Robot. Comput.-Integr. Manuf. 61 (2020) http://dx.doi.org/10.1016/j.rcim.2019.101843.

[62] F. Kong, F. Du, D. Zhao, Station-viewpoint joint coverage path planning towards mobile visual inspection, Robot. Comput.-Integr. Manuf. 91 (2025) 102821, http://dx.doi.org/10.1016/j.rcim.2024.102821, URL https://www.sciencedirect.com/science/article/pii/S073658452400108X.

[63] R.J. Fowler, M. Paterson, S.L. Tanimoto, Optimal packing and covering in the plane are NP-complete, Inform. Process. Lett. 12 (1981) 133–137, URL https://api.semanticscholar.org/CorpusID:5165774.

[64] D.S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, J. ACM 32 (1) (1985) 130–136.

[65] T.F. Gonzalez, Covering a set of points in multidimensional space, Inf. Process. Lett. 40 (4) (1991) 181–188.

[66] S. Hornus, Detecting the intersection of two convex shapes by searching on the 2-sphere, Comput. Aided Des. 90 (2017) 71–83, http://dx.doi.org/10.1016/j.cad.2017.05.009, URL https://www.sciencedirect.com/science/article/pii/S0010448517300805, SI:SPM2017.

[67] Q. Huangfu, J.J. Hall, Parallelizing the dual revised simplex method, Math. Program. Comput. 10 (1) (2018) 119–142.

[68] Industrial Path Solutions A.B., Industrial path solutions, 2024, URL www.industrialpathsolutions.com.

[69] E. Åblad, D. Spensieri, R. Bohlin, J.S. Carlson, Intersection-free geometrical partitioning of multirobot stations for cycle time optimization, IEEE Trans. Autom. Sci. Eng. 15 (2) (2018) 842–851, http://dx.doi.org/10.1109/TASE.2017.2761180.

[70] E. Åblad, D. Spensieri, R. Bohlin, J.S. Carlson, A.-B. Strömberg, Spatial–temporal load balancing and coordination of multi-robot stations, IEEE Trans. Autom. Sci. Eng. 20 (4) (2023) 2203–2214, http://dx.doi.org/10.1109/TASE.2022.3214567.

[71] D. Spensieri, J.S. Carlson, F. Ekstedt, R. Bohlin, An iterative approach for collision free routing and scheduling in multirobot stations, IEEE Trans. Autom. Sci. Eng. 13 (2) (2016) 950–962, http://dx.doi.org/10.1109/TASE.2015.2432746.