Multi-Attribute Task Sequencing Optimisation with Neighbourhoods for Robotic Systems

by
Eng. Ferdinando Vitolo
Department of Industrial Engineering

Research Doctorate
Industrial Engineering – XXIX cycle

at
University of Naples Federico II

Coordinator of the PhD School: Prof. Eng. Michele Grassi
University of Naples Federico II, School of Engineering

Supervisor: Prof. Eng. Stanislao Patalano
University of Naples Federico II, School of Engineering

February 2017

Multi-Attribute Task Sequencing Optimisation with Neighbourhoods for Robotic Systems

Method 1: Enhanced Heuristic with Hierarchical Clustering (EH^2C)

Method 2: Augmented Enhanced Heuristic with Hierarchical Clustering ($A\text{-}EH^2C$)

a Federica

ACKWNOLEGMENTS

Il primo pensiero va al mio tutor, il prof. Stanislao Patalano che, in questi anni, mi ha guidato ed incoraggiato credendo nelle mie capacità. La sua onestà e dedizione sono un esempio per me. Grazie per avermi trasmesso il senso critico (equilibrato) necessario per essere un buon ricercatore.

Grazie al prof. Antonio Lanzotti per avermi accolto nel gruppo di ricerca dandomi la possibilità di lavorare in un ambiente stimolante e amichevole.

Desidero ringraziare tutti i colleghi e amici del gruppo di ricerca con cui ho condiviso gli ultimi tre anni: prof. Giuseppe Di Gironimo, prof. Massimo Martorelli, prof. Fabrizio Renno, Stefano Papa, Andrea Tarallo, Rocco Mozzillo, Luigi Calvenese e Paola Muratto. Un abbraccio particolare a tutti i "commensali".

Un pensiero a Domenico Marzullo con cui ho condiviso questa esperienza di dottorato; in questo percorso ho trovato un vero amico.

Un grazie particolare va al "temporeggiatore", mio fratello (di vita), la persona con cui mi confronto e a cui mi ispiro. Grazie per avermi spronato e criticato quando era necessario. Sono certo che dirà: "Dunque…avete concluso." Grazie a voi (moglie inclusa) per avermi accolto; con voi mi sento a casa.

Ringrazio mio padre, mia madre e mia sorella per avermi reso quello che sono. Grazie al vostro sostegno e sacrificio ho avuto la possibilità di raggiungere anche questo risultato.

Federica, questo è un altro traguardo che raggiungo al tuo fianco. Grazie per avermi sempre compreso e capito, per avermi supportato e sostenuto in tutte le scelte e per avermi regalato le gioie più grandi. Non potevo sperare di meglio per me.

ABSTRACT

Modern manufacturing processes have to be continuously updated to catch up with fast-evolving requirements, as dictated my competitive and dynamic markets, which demand high product variety. Indeed, in the era of smart factories and cyber-physical production systems (CPPS) we are experiencing a fast transition from mass production to mass customisation. Key Enabling Technologies (KETs) are then necessary to hinge business and market needs on digital solutions which enable the rapid delivery of new and innovative products. If on one side mass customisation imposes high level of product variety, on the other hand customers wish to receive high quality products, which reflect the need for near-zero defects manufacturing systems. Therefore, the combination of macro-level changes (product variety) and micro-level variety (product defects) leads to the concept of self-evolving production systems, one of the KETs to enable CPPS. In this context, industrial robots play a key role to deploy automation and fast responsiveness.

Currently, robots are programmed following off-line methods. Tough those methods are still a premium solution to model and simulate production systems, they suffer the capability to incorporate dynamic changes. Therefore, it is crucial to introduce the new concept of dynamic robot programming which enables real-time robot adjustments. Robot programming usually consists of four steps: (1) task planning; (2) task sequencing; (3) path planning and (4) motion planning. These steps are strictly coupled although robot trajectory is mainly affected by defined tasks. In literature, task sequencing is modelled as Travelling Salesman Problem with Neighbourhoods (TSPN). There exist several methods for solving TSPN, but no one enables the dynamic programming.

This thesis aims to develop robot tasks sequencing methodology with the ultimate goal of finding the near-optimum task sequence, by minimising computational time to enable dynamic robot programming in the case of multiple and coupled tasks' attributes.

The thesis introduces two methodologies: (1) "Enhanced Heuristic with Hierarchical Clustering" (EH2C); and, (2) "Augmented-EH2C" (A-EH2C).

EH2C is a general framework to solve TSPN-like problems. The method uses a novel approach which hinges on the key idea of pre-computed feasible robot poses based on

analytical formulation of Euclidian weighted functions. Results and benchmarking studies have showed that this approach allows to reach a faster convergence rate, when compared to the top-1method available in the public domain.

The EH2C methods has been then deployed to solve robotic task sequencing problem, with multiple attributes. This has led to the A-EH2C method, which introduces the concept of multi-attribute task sequencing, as a paradigm to solve coupled and hierarchical robotic task sequencing and path planning problems.

The thesis poses the following contributions: (1) enhanced heuristic approach based on Euclidian distance to define the initial guess points for constructing tour in TSPN; (2) multi-attribute approach to find the optimised task sequencing via candidate poses solving inverse kinematics in T-space; (3) break-through paradigm shift from static robot path planning to dynamic robot path to enable on-the-fly robot re-programming to facilitate product and process adjustments.

The proposed solutions have been tested in the context of automotive body assembly systems. However, results could impact a wider area, from navigation systems, game and graph theory, to autonomous systems.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

AcI	Accuracy Index
A-EH^{2}C	Augmented Enhanced Heuristic with Hierarchical Clustering
ANSI	American National Standards Institute
API	Application Programming Interface
CAD	Computer Aided Design
CETSP	Close-Enough Travelling Salesman Problem
CI	Coverage Index
CIH	Constricting Insertion Heuristic
CoI	Collision Index
CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
C-space	Configurations space
DOF	Degree of Freedom
DTD	Device Task Definition
EE	End-Effector
EH^{2}C	Enhanced Heuristic with Hierarchical Clustering
FoV	Field of View
GA	Genetic Algorithm
GRASP	Greedy Randomised Adaptive Search Procedure
GTSP	Generalise Travelling Salesman Problem
GTSPN	Generalise Travelling Salesman Problem with Neighbour
JARA	Japanese Robot Association

JIS	Japanese Industrial Standards
MINLP	Mixed Integer Non-Linear Programming
MTP	Multi-Goal Path Planning Problem
OLP	Off-Line Programming
RBA	Rubber-band Algorithm
RI	Reachability Index
RIA	Robotics Institute of America
RPP	Robotics Path Planner
RRS	Realistic Robot Simulation
TCP	Tool Centre Point
TPP	Touring a sequence of Polygons Problem
TSP	Travelling Salesman Problem
T-space	Tasks space
TSPN	Travelling Salesman Problem with Neighbour
TSP-ND	Travelling Salesman Problem with Neighbourhoods and Durative visits

1 INTRODUCTION

1.1 Motivation

The race to improve goods and process performances has pushed modern manufacturing to re-think and re-structure production systems. This has laid down a disruptive paradigm shift: from *automated* to *smart & connected* systems. In Germany, referring to these paradigm, on the reminiscence of software versioning and inspired by future expectations, it was conceived the term “Industrie 4.0” [1] better-known as “Industry 4.0”.

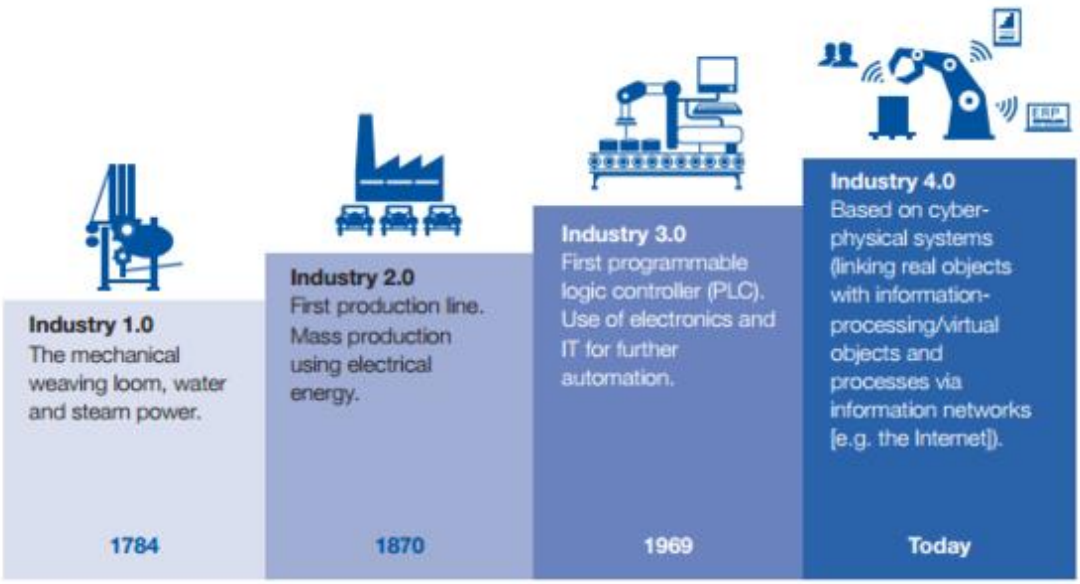

Figure 1 – Chronology of industrial revolutions [4].

Industry 4.0 refers to the concept of cyber-physical systems (CPSs) [2], which entails three mayor ingredients: (1) machinery (physical) with smart sensors; (2) data connectivity; (3) model (cyber) representation of the physical systems. Though the

concept of "digital" and "physical" integration has been investigated for years in engineering and computer science, only over the last decade the IT technology has sufficiently evolved to make possible the seamless integration.

Drath and Horch [3] provide a very clear example of CPS:

"…*traffic lights today either act independently from each other or are controlled by a central traffic control system. As a CPS, the physical traffic lights would have an object representation in the network providing their current color and time schedule. Based on these data, future cars could inform themselves about the plan of the next traffic light, adjust speed, or provide automatic motor on–off features to minimize emissions. Future navigation systems could calculate an optimal route through traffic for every car, dependent on its position, destination, and other related information, such as traffic jams. Once cars feed their position, speed, and destination back into the network, the traffic lights could orchestrate and optimize their behavior with respect to an optimal traffic flow. Police, ambulances, or fire engines could control green lights for optimal security and safety in the city*".

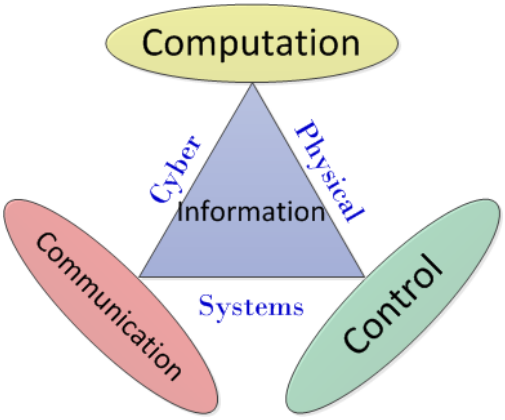

Figure 2 – Concept of cyber-physical systems [5].

The application of the CPS concept to the production systems has brought the definition of cyber-physical production systems (CPPS) [3], also known as smart factory or connected factory, which consists of physical machineries interconnected and identified in a factory network where they can be simulated an optimised to improve production making it faster, more flexible and efficient having higher-quality goods at reduced costs [6]. Therefore, the industrial vision of future production predicts smart products which control and optimise themselves in their manufacturing process [7].

One of the most used and implemented machines within production systems are industrial robots, which are adopted by automotive, aerospace, electronics, appliance, chemical, plastics and rubber and material industry. For instance, total worldwide stock operational industrial robots at the end of 2015 was about 1.6 million units (increased by 11% compared to 2014) estimating an increase by at least 13% on average per year up to 2019. The market value in 2015 was estimated to be US$35 billion (increased by 9% compared to 2014) [8].

One of the largest industrial robot installations is the automotive sector. Automotive industries widely use multi-stage assembly systems consisting of multiple machine stations/stages to obtain the final product [9]. As each stage is composed by one or more robots which repeat the same task multiple times, it is important to optimise robot path in order to minimise execution cost in terms of cycle time or energy. Optimal path is obviously related to a specific sequence of tasks. In classical robot path planning, tasks are *statically* specified. For example, for welding robots, seams' locations are defined using design data and product performances, such as strength and stiffness. Static robot path planning underlays the assumption of ideal product and processes. However, it has been proved that most of the changes occurring after design release are imputed to dimensional and geometrical variations [10]. This leads to the need of dynamically re-program and re-root tasks to automatically reconfigure the robotic system. For example, in case of measurement and inspection stations, in case root cause of defects cannot be isolated and localized based on the data gathered by a *static* measurement systems, the measurement systems should be reconfigured by selecting additional measurement points to increase likelihood of isolating the root cause. The scenario of dynamic re-programming is also described in [11] where authors introduce the concept of moving goals. Therefore, because the process is constantly changing, adaptive control system should be adopted, implying that robots should be adaptively re-programmed.

How in the traffic light example [3], CPPS shall be able to predict production behaviour and perceive the production fluctuations to automatically self-recover and adjust to fit on-the-fly the production variations.

Robot tasks are programmed following two major methods: (i) **on-line** - robot movements are manually recorded in teaching mode and executed in production mode;

(ii) **off-line -** automatic path planning programming through simulation based on CAD model [12].

Tough off-line programming (OLP) has the premium benefit of developing the optimum robot task planning, it is based on a *static system* assumption which is incompatible to dynamically react to fluctuations and variations [13]. This brings the new concept of *dynamic robot programming* which enables real-time robot adjustments.

Defining the robot tasks entails multiple and coupled **attributes**, which can be summarised as follows:

- *cycle time* (A1) – to minimise execution time of multiple tasks;
- *energy consumption* (A2) – to minimise energy consumption in robot transitions;
- *path length* (A3) – to minimise length of the end-effector's path

Either (1), (2) or (3) are strongly correlated to:

- *pose quality* (A4) – accuracy and repeatability of the end-effector pose (both position and orientation)
- *collision* (A5) – robot movement must be collision free and avoid
- *robot placement* (A6) – robot placement (both position and orientation) with respect to the workpiece. Robot placement is directly related to the *accessibility of tasks.* For instance, it may happen that the same task though feasible in terms of collision and pose quality, can be executed by multiple paths (multiple accessible paths). As consequence, there is no guarantee of reaching the minimal cycle time attribute.

As finding the exact solution for the robot transition movements is computational hard, some approaches decompose the problem in sub-problems. Kolakowska et al. [15] decomposed the problem as follows: (1) task planning; (2) task scheduling; (3) motion planning. Combining what is defined in [14] and [15], classical resolution approach is based on a 4-step decomposition approach:

1. *task planning* (step 1) - tasks are described in a well-defined coordinate system
2. *task sequencing* (step 2) - sequence of tasks is generated according to attributes. It is of interest to notice that classical task sequencing computes (near) optimal sequence in the Cartesian space, also called T-space. This implies that robot configurations are not accounted, but only end-effector positions.

3. *path planning* (step 3) - inter-tasks route is generated according to task sequence and attributes. It allows to computes both the position and orientation of the end-effector. The computation is performed into the configuration space of the robot, also called C-space.
4. *motion planning* (step 4) – robot movements are generated to follow the path.

Steps (2) to (4) are linked to each-other and the problem becomes NP-hard [16]. In fact, step (2) computes the optimal task sequence ignoring the end-effector orientation, which is calculated in step (3). This implies that some of the solutions computed in step (2), even though guarantee the (near) optimal sequence criterion, might fail to satisfy the robot accessibility criterion.

Proposed methods following the 4-step decomposition approach are heuristics-based. They mostly re-iterate step (2) to (4) until satisfying attributes A1 to A6. The integration mostly follows a brutal sequential approach which allows to reach optimal solutions, however it suffers the number of unnecessary re-iterations.

1.2 Objectives

Industrial robots have to perform several tasks, moving among configurations arranged for each task and avoiding collisions. A representative scheme is depicted in Figure 3. It is quite important as well as very complex to determine the optimal sequence of tasks visited by the representative point of the end-effector - also called Tool Centre Point (TCP) - to generate the optimal robot's movements through tasks.

Classical approaches consider a simplified formulation: task sequencing and path planning problem are completely or partially decoupled. Task sequencing is solved in T-space. Path planning deals with the robot configuration for each end-effector pose and the sequence of configurations that moves the robot among configurations; it is solved in C-space. As no robot information is involved in task sequencing, no feasible solutions are guaranteed. Therefore, these methods require some zigzagging iterations between task and path to converge on a solution.

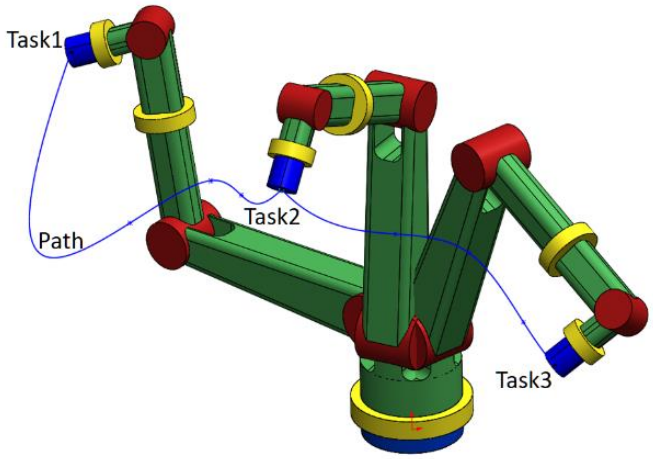

Figure 3 – Robot performs three tasks arranging three different configurations.

This dissertation focuses on robot tasks sequencing and aims to develop a novel methodology to find near-optimum solution, by minimising computational time to enable dynamic robot programming in the case of multiple and coupled tasks' attributes. The proposed methodology is based on the two steps:

1. integrated **task planning** which entails (i) robot placement and (ii) tasks' definition (introduced in); and,
2. **augmented task sequencing** as a measure of the enhanced T-space with pre-computed feasible configurations. It is of interest to notice that classical *task sequencing* only focuses on the T-space and subsequent robot configurations, along with their feasibility, are only computed in the later path planning stage. The proposed method allows to take into account the feasibility of the robot configurations from the very early stage of the optimisation workflow. This leads to the following two benefits: (i) the computed solution is (near) optimal and feasible; and, (ii) reduction of costly forward and feedback iterations between task sequencing and path planning.

1.3 Research Questions

The dissertation addresses the following questions:

1. How to find the optimal task sequence and poses integrating robot information in T-space

2. How to develop efficient method to enable seamless integration of task sequencing and path planning
3. How to upscale the integrated task sequencing and path planning model to multi-attribute scenarios
4. To demonstrate the proposed solution to automotive door assembly systems

1.4 Contribution

The dissertation introduces the concept of multi-attribute task sequencing, as a paradigm to solve coupled and hierarchical robotic task sequencing and path planning problems. Contributions are summarised as follows:

(1) Enhanced Travelling Salesmen Problem with Neighbourhoods (TPSN) to solve the task sequencing problem. The dissertation proposes to use an enhanced heuristic approach based on Euclidian distance to define the initial guess points for constructing tour in TSPN;

(2) Multi-attribute approach to find the optimised task sequencing via candidate poses solving inverse kinematics in T-space;

(3) Break-through paradigm shift from static robot path planning to dynamic robot path planning; and,

(4) Capability to implemented proposed approach to enable on-the-fly robot re-programming to facilitate product and process adjustments.

1.5 Thesis Outline

The reminder of the dissertation is as follows:

Chapter 2 – Background. It lays down the common terms and concepts used in industrial robotics.

Chapter 3 – Related Works. It reviews state of art and identifies current trends and limitations.

Chapter 4 – Proposed Methodology Overview. It presents the Enhanced Heuristic with Hierarchical Clustering method for TSPN and its comparison with well-known methods

Chapter 5 – Augmented EH^{2}C for Task Sequencing Optimisation. It presents the multi-attribute method for robotic task sequencing and a case study.

Chapter 6 – Conclusions and Further Works. It draws final remarks and potential future development.

2 BACKGROUND

This section defines common terms and concepts used for industrial robotics. Besides, it explains the robot path planning principles.

2.1 Definitions

What is a robot? Nowadays it is very difficult to give a thorough definition of robot. Broad associations (like the Robotics Institute of America (RIA) and the Japanese Robot Association (JARA)) and Standardisation Institutes (like American National Standards Institute (ANSI), Japanese Industrial Standards (JIS) and International Organization for Standardization (ISO)) have tried to give a definition and classification of robots. A robot is "*a reprogrammable, multifunctional manipulator designed to move materials, parts, tools or specialised devices through various programmed motions for the performance of a variety of tasks*" [17].

The widespread industrial robot is the articulated/anthropomorphic robot belonging to the manipulating type. As defined in [18]: "*A manipulating industrial robot is an automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications*." In this thesis, we focus on the articulated robot but any assumptions and results can be applied to any industrial robot type.

An industrial robotic arm is usually composed by a base, a sequence of links (rigid bodies) and an end-effector (EE) connected by kinematic pairs (joints), see Figure 4. These components define the kinematic chain (sequence of links connecting the two ends of the chain: base and end-effector). The common industrial robot has got six revolute joints: ϑ_1 – waist; ϑ_2– shoulder; ϑ_3– elbow; ϑ_4 – wrist rotation; ϑ_5 – wrist bend; ϑ_6 –

flange rotation. The number of joints determines the number of Degrees of Freedom (DOFs).

We define the TCP as the EE point of interest to be tracked; it can or cannot belong to the EE tool as well as exist geometrically because defined by functional parameters. The TCP position (x_{EE}, y_{EE}, z_{EE}) and orientation (α, β, γ) compose the pose of the TCP.

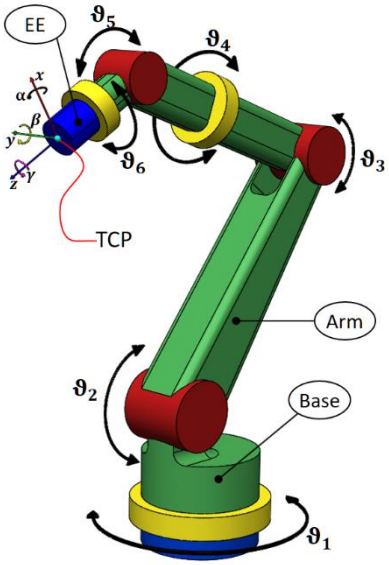

Figure 4 – Robot components: base, arms and end-effector. They are linked by planar revolute joints (red cylinders) and orthogonal revolute joints (yellow cylinders).

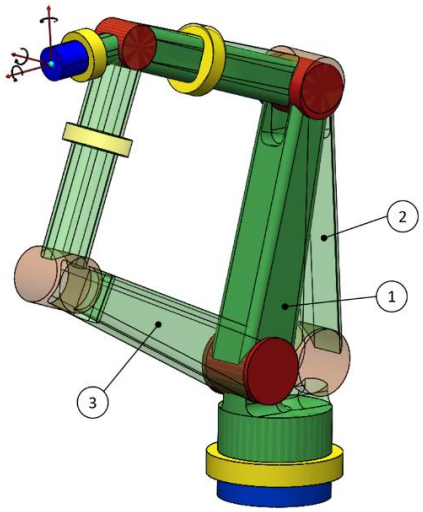

Figure 5 – Example of three configurations of a redundant robot. (1) and (3) are named elbow-up and elbow-down respectively.

The main coordinate reference frame is the robot triad placed in the "base". Henceforth, we will refer to it as robot reference. Starting from the robot reference, a pose can be defined in two ways, in two different space:

- in the task space or Cartesian space (T-space) by coordinates $(x, y, z, \alpha, \beta, \gamma)$;
- in the configuration space or joint space (C-space) by joint configurations $(\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6)$.

T-space is only related to the EE (it defines the pose) while C-space introduces information on the robot and its links (it defines the robot configuration). Therefore, each pose can be reached by multiple robot configurations (a typical industrial robot, with 6 DOFs, could arrange up to 8 configurations for a defined pose – see Figure 5).

Moving from C-space to T-space is called *forward kinematics* as to each robot configuration correspond only one pose. Conversely, moving from T-space to C-space gives multiple solutions and it is called *inverse kinematics*.

Robots perform any specific job following a path which robot movements correspond to. A job consists of several tasks. There are two types of paths: (1) task or effective path which is the robot path to accomplish that specific task; (2) supporting or inter-tasks path which is the robot path that connects tasks [15]. Therefore, a job path consists of task and inter-tasks paths.

The route that leads from a robot configuration to another is named path. A path is the geometrical description of the robot motion, i.e. locus of points; a trajectory is a path with a motion law [19]. Here, we will refer to the path as the locus of TCP points over time. All TCP accessible points are contained in a volume named "robot workspace".

2.2 Commercial Tools

This Section introduces the existent commercial solutions, hardware and software, for the task sequencing and path planning. Pros and cons are identified and critically discussed.

2.2.1 **Robot Studio**

Robot Studio is an ABB's PC-based robot programming software (see Figure 6). The ability to program a robot in the virtual world before it operates in the real world has dramatically changed the way companies and individuals think about programming robots. Over the last decade it has become an increasingly popular way to test robot operation before a mistake on the factory floor results in damage, stoppage and/or loss of money. The traditional method of programming robots, using a Flex Pendant attached to the robot controller, works well for some tasks, but robots have been placed into ever more intricate and complicated operations and even the most skilled human programmer staring at a screen full of countless lines of code would be hard pressed to accomplish.

Once the program is completed in the virtual world it can simply be downloaded straight to the robot controller in the real world, and as long as everything in the real world is set up exactly as it was in the virtual world, the program will run exactly like it did on the PC.

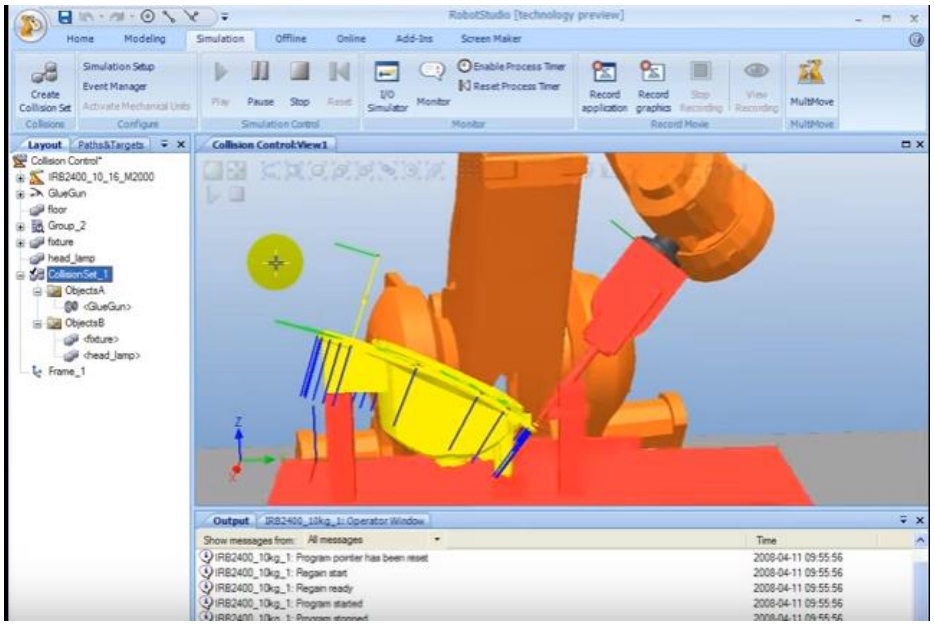

Figure 6 – Robot Studio collision detection

Robot Studio allows to check reachability, avoid collision and detect singular issues. Robot Studio has several functions for testing how robots reach and move to targets. They are useful both for finding the optimal layout when building a station and during programming. With Robot Studio we can detect and log collisions between objects in the

station. A collision set contains two groups, Objects A and Objects B, in which we place the objects to detect any collisions between them. When any object in Objects A collides with any object in Objects B, the collision is displayed in the graphical view and logged in the output window.

After having created a collision set, Robot Studio checks the positions of all objects and detect when any object in Objects A collides with any object in Objects B. Activation of detection and display of collisions depend on how the collision detection is set up. If the collision set is active, Robot Studio checks the positions of the objects in the groups, and indicate any collision between them according to the current colour settings.

After detecting a collision, we can modify the path of the robot's tool and run the program again to check whether there are collisions also with the new setup. If now collisions are avoided, this new path is saved as a collision-free path for the robot.

2.2.2 DELMIA

DELMIA is the Digital Manufacturing and Production Solution of Dassault Systèmes, optimising production systems and processes (see Figure 7). DELMIA Device Task Definition (DTD) delivers the capability to program and simulate forward kinematic mechanical devices, ranging from simple clamps to complex lift-assist mechanisms. It also provides the ability to manage multiple devices, integrate them within the V6 3D work-cell layout, and perform feasibility studies. Each device is individually programmed with tasks that are sequenced and simulated to eliminate any interference and obtain optimal cycle times.

DELMIA Device Task Definition provides an interactive V6 3D environment which allows users to define the tasks for each device in the context of the shop floor. Users are able to sequence the tasks of each individually programmed device in order to achieve synchronised motion between the devices in the work-cell.

Single or multiple device tasks can be simulated in 3D to locate and correct any interferences or collisions in the work-cell. Users can evaluate and optimise device activities to achieve desired cycle times.

The robot programmer can automatically optimise the robot's motion by computing standard motion parameters such as turn numbers, configuration, gantry and rail values along a robot trajectory[20]. It also provides tools which optimise cycle time and reach to create a collision-free path.

Moreover, DELMIA Robotics Path Planner (RPP) provides tools for automatically computing collision-free and optimised trajectories for industrial robots. Robotics Path Planner provides a highly-efficient command for automatic collision-free path planning to facilitate robotic feasibility studies and off-line programming.

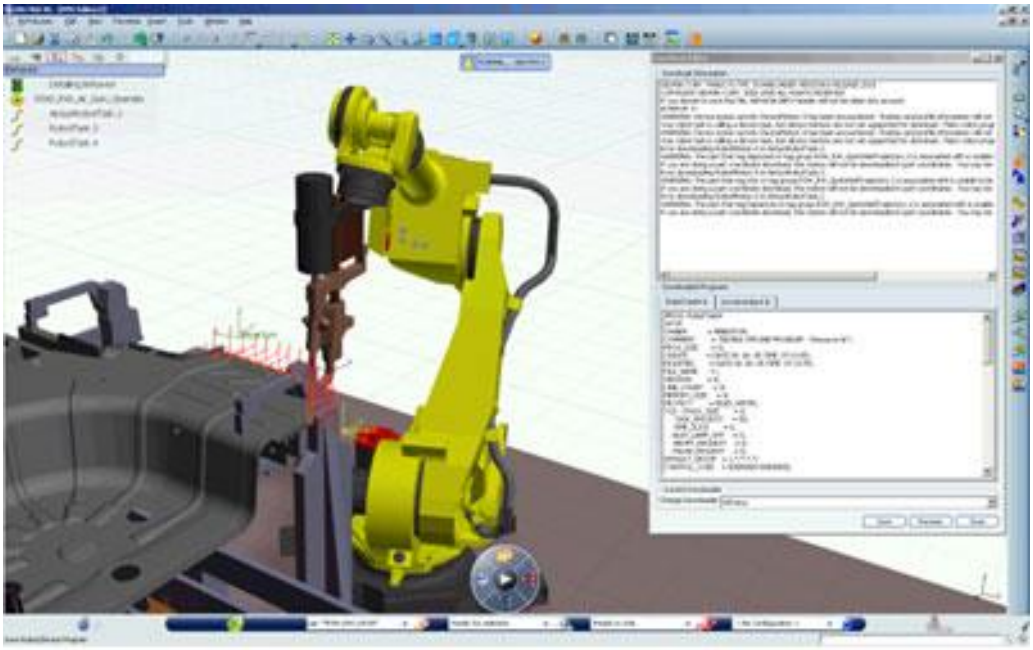

Figure 7 – Example of a robot simulation with DELMIA

Cycle times are minimized by RPP by optimising automatically new trajectories calculated to fit exactly each new project. Path of the tool centre point frame for linear motion, or path in the configuration space for joint motion, is minimized with better cycle times than can be achieved by other methods. By applying RPP to DELMIA robot task motion activity, RPP creates collision-free and optimised DELMIA motion activity.

Robotics Path Planner automatically transforms a robot task, updating a motion activity with potential collisions between the robot and its environment into a collision-free trajectory. When the robot and its environment needs to be modified and updated the previously defined task can be automatically recomputed providing a fast versioning

check and task update. Once a trajectory is computed, it can be optimised to reduce the robot cycle time. The non-trivial task of robot configuration space optimisation is achieved automatically by RPP. The resulting joint motion interpolation yields a faster motion and a lower risk of singularity.

2.2.3 RobCAD

RobCAD is a Siemens PLM Software for robotic work-cells verification and off-line programming. Tecnomatix RobCAD software (see Figure 8) enables the design, simulation, optimisation, analysis and off-line programming of multi-device robotic and automated manufacturing processes in the context of product and production resources. It provides a concurrent engineering platform to optimise processes and calculate cycle times. With RobCAD, you can design life-like, full-action mock-ups of complete manufacturing cells and systems. RobCAD enables manufacturers to flawlessly introduce automated processes by allowing manufacturing engineers to virtually validate automation concepts upfront.

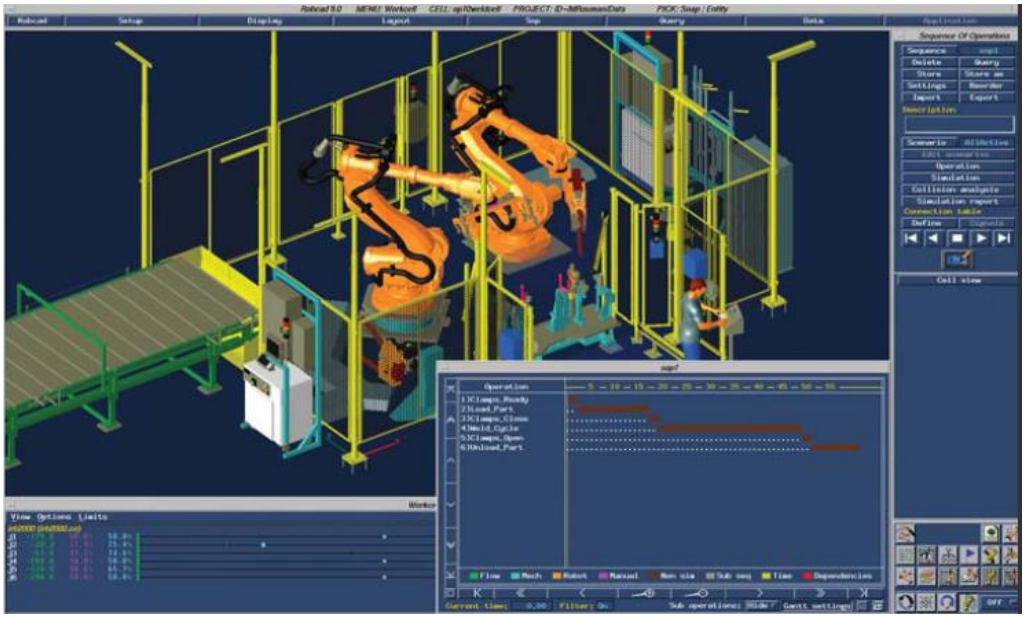

Figure 8 – Robcad software for off-line programming

RobCAD generates configurable motion plans based on the controller features. It allows calculation of cycle times, analysis of real-time performance and saves testing

time. The RRS (Realistic Robot Simulation), which is based on using the real controller motion planning software, offers extremely accurate cycle time calculation.

Robcad off-line programming enables accurate simulations of robot motion sequences and the delivery of machine programs to the shop floor. Moreover, RobCAD can dynamically detect collisions during robot simulation and motion, preventing costly damages to equipment. In fact, for automatic path planning**,** RobCAD generates collision-free robot and part assembly paths by using automatic path planning technology.

2.2.4 Kineo

Kineo is a Siemens PLM software. Kineo solutions include advanced software components and standalone applications for automatic motion/path planning and collision detection (see Figure 9). Kineo products satisfy a wide range of virtual prototyping requirements, from assembly or disassembly clearance validation to collision-free robot applications. In modern end-user CAD, CAM, CAE, 3D digital mock-up and robotics systems, these productivity tools help automate path planning and clash detection factors which in turn save customers time, costs and resources.

Figure 9 – Example Kineo software

Kineo components provide leading path planning and collision detection tools for CAD/CAM applications. In autonomous robotics, Kineo-based path planning and collision detection maximize the operational efficiency of robotic systems.

Kineo Collision Detector allows to check spatial interferences, or collisions, between hierarchical assemblies of triangle mesh surfaces, or polyhedrons. Kineo Collision Detector can be used to perform different kinds of interference analyses, including:

- Exhaustive Boolean check to determine if analysed objects are colliding and, if so, reports every pair of colliding triangles;
- Exact distance to determine if analysed objects are colliding and, if not, reports the shortest distance between them; and,
- Penetration to determine if objects are colliding and, if so, reports a translation vector that suppresses the collision.

Every object can have its own tolerance value, which is the size of a clearance zone added around the object. Kineo Collision Detector is optimised for low response times, with a built-in multithread capability, enabling the best hardware performance. Thanks to its stateless, thread-safe mode, Kineo Collision Detector is suitable to run different tests over the same scene in simultaneous threads. This offers new possibilities to multithreaded applications aimed at performance and reactiveness. Instead of waiting for tests to return, the process can span new tests in new threads and use all available computing power.

2.3 Summary and Remarks

Off-line programming is useful tool for saving money and time when designing a new work cell. Simulation and OLP allows designers to study multiple scenarios of a work cell and potential failures can be validated in advance and corrective actions generated accordingly.

Commercial and academic/open source robotic software for OLP mostly focuses on the motion planning optimisation, and neglect optimisation of the task sequencing. All robot targets are programmed by the operator that should simultaneously consider optimal EE placements, reachability and sequence constraints.

Furthermore, they optimise the path or motion planning on a fixed sequence of tasks. For example, after defining the TCP locations, RobCAD computes the connection path generating additional via-location and then check the reachability. RobCAD optimises path by adjusting just via-location [22]; KUKA Sim works in the same way [23], Kawasaki Pc-Roset omits the task sequence optimisation too [25]. DELMIA V5 doesn't include sequencing in path optimisation but can be partially developed using APIs as in DELMIA V5 Robotic Drilling Application.

Although existing solutions for OLP and path planning are standard toolkits in modern design architectures, developed solutions are not able to modify and choose a new path automatically and to dynamically react to fluctuations and changes, has happening in real-life production systems.

3 RELATED WORKS

This chapter highlights research key topics. It analyses related works illustrating their characteristics and limitations in Robotics application.

3.1 Introduction

Although path optimisation concerns both task and inter-tasks paths, they can be computed in separate way considering that the input and output configuration of the effective path can be considered as two different configurations in the supporting path. Hereinafter, supporting path will be referred as path.

The robot paths have to efficiently avoid collisions and unnecessary movements. The path planning problem aims to find the sequence of the robot configurations to accomplish the job. Robot systems can reach a given location assuming several configurations, ideally, infinite; therefore, the sequence of tasks is affected by multiple attribute and objective function; optimisation cannot neglect them.

3.2 Robot Programming

Solutions for robot programming aim to generate the optimal trajectory to perform a specific job in production. Optimal trajectory is affected by multiple attribute: pose quality, robot redundancy, collision and robot placement. Since robot has multiple solution for each EE-pose in T-space, robot configurations need to be selected by considering simultaneously the attributes assessing reachability, minimising configuration transition, avoiding collision and evaluating pose performance. Although there are a lot of researches and commercial OLP tools, most of them are not able to

provide an optimal robot path automatically but a manual assignment or an adapted (APIs) application is necessary [12].

3.3 Robotic Task Sequencing

Sequence of robot tasks are classical tour-searching combinatorial problems modelled as the Travelling Salesman Problem (TSP) [27][28]. This is a well-known problem of tour-searching in many branches of mathematics, operations research and computer science; it aims to find the shortest tour among a given set of points visiting each point exactly once and returning to the original one (Hamiltonian cycle). In this case, each task is formulated as point. This formulation is not good for multiple inverse kinematics because robot can arrange multiple configurations for the same placement as well as multiple position to accomplish the same operation. Therefore, TSP can provide only an initial tour approximation [29].

Due to multiple configurations, it is more realistic formulate the problem as a point set which corresponding several robot configurations or EE placements. This formulation is named Generalised TSP (GTSP) [30]. GTSP is obtained substituting each single point with a cluster of points, the shortest path visit one point for each cluster. GTSP is applied in [31] where the authors generate the clusters by sampling a set of configurations for each location. This solution is always limited among the sampled points. An interesting link between GTSP and robotics was established in [32], where the authors introduce the multi-goal path planning problem (MTP) where a cluster of poses are modelled as a cluster of points.

3.3.1 Traveling Salesman Problem with Neighbourhoods: Introduction

Despite GTSP partially overcome the limitations on multi-inverse kinematics (multi-IK), it requires a certain discretisation that means errors in the final solution and a partial task volume representation. GTSP solution quality improves with augmenting point numbers increasing computational complexity.

To obtain an acceptable optimised solution the search space has to be continuous, i.e. a region. When the points change in regions (examples: areas in 2D; volumes in 3D) the

TSP becomes Traveling Salesman Problem with Neighbourhoods (TSPN) [33] where each region is visited once. This problem consists of finding an optimal sequence and an optimal point within the neighbourhoods. This can be solved by minimizing the path length among neighbourhoods [34] [35] [36]. Gentilini et al. [37] use TSPN to formalise the path optimisation but they neglect the multiple inverse kinematics. Some researches formalise the multiple configurations for each location goal as a cluster of regions. This increases the computational complexity.

As for TSP, when the region is substituted by a cluster of regions the problem becomes generalised: Generalised TSPN (GTSPN) [38]. Vicencio et al. [39] use GTSPN to optimise a six-rotor path planning to overcome the limitations of the TSPN formalisation. A recent survey on task sequencing problem [40] collects and classifies the main implementation pointing out that the TSPN heuristic solvers are focus on $\mathbb{R}^2$ and $\mathbb{R}^3$ space. TSPN solvers are not limited by the space rather by currently formulation that cannot allow to integrate path and task [41].

3.3.2 Traveling Salesman Problem with Neighbourhoods: Modelling

TSPN aims to find the shortest path via regions visiting each region once. This formulation allows to optimise both sequence and location within neighbourhoods.

Arkin and Hassin [33] first introduce TSPN studying an approximation algorithm to solve it. Later, other researchers ([36], [42] among others) face TSPN in approximation domain. Over the time, interesting in TSPN from application technologies generated a new requirement: solve TSPN faster. This has lead researchers to develop efficiently TSPN heuristic algorithms.

Mennell [43] faces the Close-Enough TSP (CETSP) that is a TSPN-like problem where regions are assumed as disk. His proposed approach splits the main problem in TSP and Touring a sequence of Polygons Problem (TPP) [44]. The approach provides a decomposed approach based on two steps: (i) find an initial sequence; (ii) improve solution. It first finds a sequence among disk by a TSP solver assuming disk as centre points; then it optimises point inside disk by TPP solver. The approach's weakness is nestled into initial point's selection. On the same decomposition principle, Elbassoni et

al. [36] face TSPN providing an approach that first optimise the point using a Euclidean metrics and then find the optimal sequence. The same decomposition idea driven Alatartsev et al. [45] which proposed the Constricting Insertion Heuristic (CIH) method as an efficient way to solve TSPN based on a simultaneous Mennell-like approach (TSP + TPP).

Gentilini et al. [46] first applied TSPN to model robotic task sequencing considering the neighbourhoods as a continuous domain. However, the developed method has been proved with only 16 tasks because the Mixed Integer Non-Linear Programming (MINLP) formulation proposed is computationally expensive. In [34], authors adopted the CIH algorithm to integrate task and path. Although CIH tries to solve TSPN simultaneous, it is structured in sequential way.

3.4 Robotic Path Planning

Robot path planning is an interested topic in the robotic community. The path planning problem is applied to all those applications with involve automated systems: painter robots [47]; cleaner robots [48]; spot welding robots [49]; remote laser welding robots [50]; underwater inspection vehicles [51]; measurement robots [52].

Path planning aim to find the optimal robot path according to multiple attribute and objectives. Path planning problem involve all robot aspects (except robot motion law) and is solved in C-space where all robot information is present.

Spitz et al. [52] proposed heuristics method based on TSP tour construction to solve CMM path planning minimising path length. They consider obstacles but neglect robot redundancy assuming just one configuration for each pose.

Gueta et al. [29] proposed a method to avoid collisions between robot and workpiece placed on rotating table. They optimise cycle time considering system redundancy to select a different configuration for collision-free path assuming straight-line path fixed in C-space. They model the problem as a cluster of configuration solved by a heuristics TSP algorithm.

It is difficult to manage C-space information because robot is described by configurations missing a clear search space representation, that why researchers prefer

solve the problem in T-space where the robot can be modelled by TCP positions (please refer to [29], [31], [53], [54]).

As path planning solution is strictly affected by task sequence, to generate the optimal sequence to get a real near-optimal path.

3.5 Integration Approaches

Wurll et al. [32] first introduced the integration concept. The authors introduce the multi-goal path planning problem (MTP) where a cluster of poses is modelled as a GTSP to find a collision-free path solved in T-space. Later, Faigl et al. [68] face MTP with regions. In general, given a set of robot goals, MTP stands to find a shortest path among goals.

The problem of task integration is presented in [56], even if the authors predefine the sequence by means sampling without consider it in the optimisation.

Zacharia et al. [53] introduce a method to simultaneously solving motion planning and task sequencing with a TSP formulation. Indeed, they deal with a fixed task-points therefore with a fixed EE positions.

Recently, in robotic remote laser welding, Kovacs [41] introduces a novel model problem: Traveling Salesman Problem with Neighbourhoods and Durative visits (TSP-ND) for task integration. He proposes a meta-heuristic approach based on Greedy Randomised Adaptive Search Procedure (GRASP) to solve TSP-ND.

3.6 Summary and Remarks

Finding optimal sequence of tasks is crucial in all industrial applications where repetitive jobs are performed. Currently, there are few approaches that allow an automatic task sequencing. None of them use a complete integration of task sequencing and path planning lead to solutions far from the optimal one.

Multi-attribute path planning with a relatively simple computation process seems be a very big challenge. Adding multiple attributes increase the search space making exact methods application difficult [40]. Researchers use mainly decomposition approaches to

reduce the problem to simple ones that can be solved sequentially or parallel applying heuristics methods to get solution in reasonable time.

A novel effort in the problem formulation is then required to provide a synergic attributes integration.

4 PROPOSED METHODOLOGY OVERVIEW

This chapter describes the proposed methodology and highlights formulation and gives an overview of the developed approach.

The proposed methodology aims to solve the task sequencing problem using the Travelling Salesman Problem with Neighbourhoods (TSPN). It optimises both sequence and via-points and it is named Enhanced Heuristic with Hierarchical Clustering (EH2C). Compared to the best methods published in the public domain, EH2C allows to reach a faster convergence rate, because it analytically evaluates the most promising via-points by solving a preliminary sub-optimisation model. Instead, existing methods generate guessed via-points by random sampling, which are not guaranteed to be sufficiently close to the sub-optimal solution.

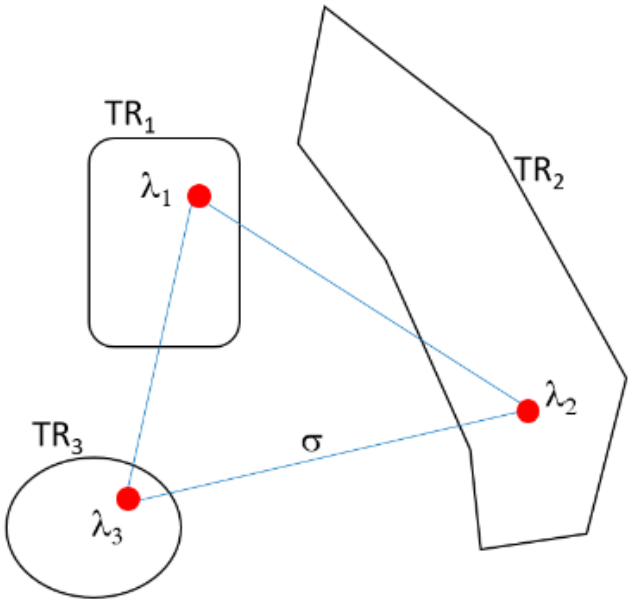

Figure 10 - Sequence of tasks

4.1 Introduction

Industrial robots perform a cycle of tasks to carry out a job. For a given task, T_i, robot can reach a pose $\boldsymbol{\lambda}_i$ to perform that task.

Given n number of tasks, for each task "i" there exist a region "TR" defined by technological parameters which characterise EE-pose for the task execution (see Figure 10). These parameters are related to specific applications. For example, technological parameters for inspection robots with optical camera system are: optimal operating distance, depth, length and width of field of view.

We aim to find the optimal sequence, $\boldsymbol{\sigma}$, of poses, for pre-defined multi-attribute, by minimising cycle time t_{min} as well as selection of the optimal pose for each task/region. Therefore, one can formally write:

$$\begin{aligned} &\forall\, T_i \,\exists\, \boldsymbol{\lambda}_i \in TR_i \;:\boldsymbol{\sigma} \;\Rightarrow t_{min} \\ &i \in [1, n] \end{aligned} \tag{1}$$

4.2 Methodology Overview & Key Principles

The proposed method aims to find an optimal task sequencing taking into account robot attributes into the T-space (see Figure 11).

A pose can be defined in T-space by means of position $\boldsymbol{P} = (x, y, z)$ and orientation $\boldsymbol{Or} = (\alpha, \beta, \gamma)$; whereas, in C-space through configuration $\boldsymbol{q} = (\vartheta_1, \dots, \vartheta_m)$ where m is joint numbers. Although in C-space it is possible to define a complete EE-pose as well as robot configuration, it is difficult to define an optimal task sequence. Therefore, it is more convenient to model the robot task sequencing problem in the T-space; therefore, the proposed approach formalises the task sequencing problem in T-space and brings attributes from C-space, with the aim of calculating both optimal task sequence and feasible poses. The reader may notice that feasible poses can only be computed in the C-space, where robot information is made available.

Given a set of n tasks, the proposed method firstly defines task regions TRs and, then, calculates minimum distances among them. Subsequently, it selects via-pose by simultaneously optimising pose-to-pose distances, pose accuracy, collision and reachability; finally, a task sequencing $\boldsymbol{\sigma}$ is generated through via-poses $\boldsymbol{\lambda}_i, \forall i = 1, \dots, n$.

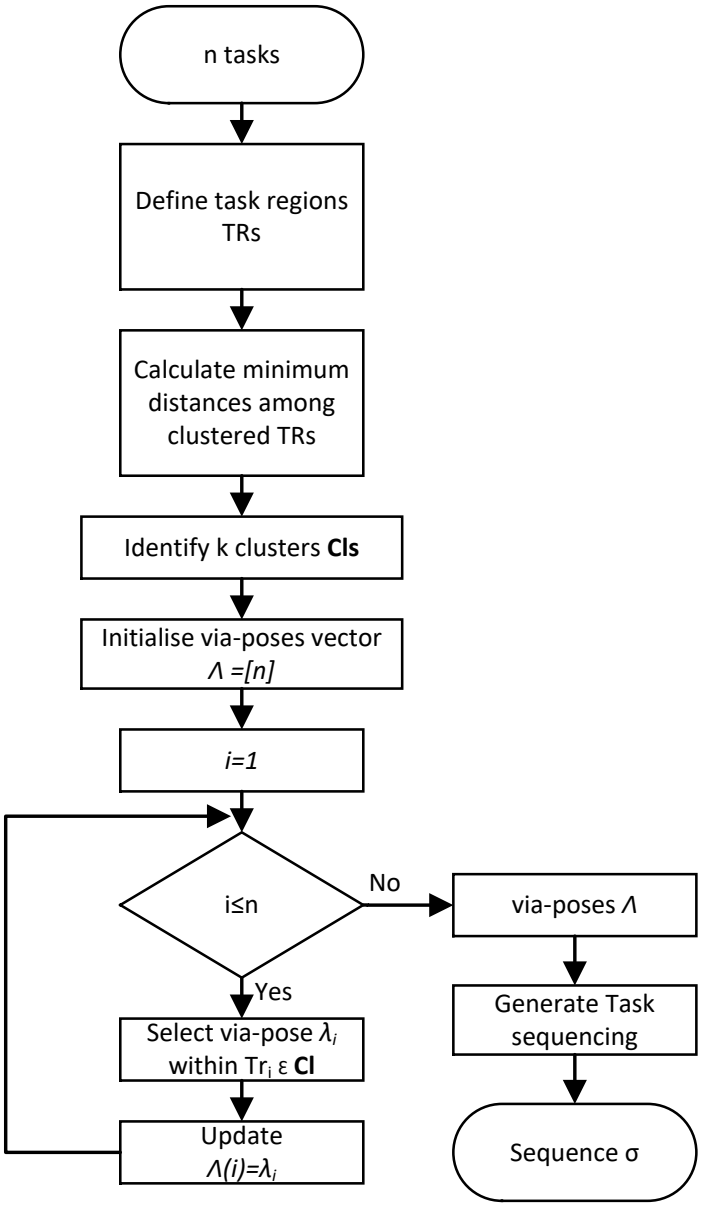

Figure 11 - Task sequencing flowchart

In the T-space, robotic task sequencing can be modelled as TSPN, where each neighbourhood (region TR) represents robot task and any inner points represent the position, $\boldsymbol{P}$, of the EE-pose.

The proposed method, named Enhanced Heuristic with Hierarchical Clustering (EH^{2}C), solves the TSPN using Euclidean distance as key metric. Then, EH^{2}C has been expanded and named Augmented EH^{2}C (A-EH^{2}C) to introduce pose orientation, $\boldsymbol{Or}$, and enable to check the feasibility of reachability, collision and pose accuracy.

4.3 Heuristic with Hierarchical Clustering

TSPN, which belongs to TSP-like problems, is a tour-searching model that generalises the TSP by substituting points for regions. Each region corresponds to the assigned robotic task. It aims to find the minimal cost cycle via regions visiting each region only once. TSPN entails two sub-problems: (1) allocate a via-point inside each region; and, (2) find the best tour through via-points.

TSPN is an NP-hard problem and exact solution is computational expensive. Classical methods use heuristics approaches. In literature, these methods are classified as follows: (1) TSP tour construction – it finds a feasible tour; (2) TSP tour improvement – it modifies exist tour to minimise its cost [28].

Mennell [43] proposed a 3-step method as follows: (1) reduce search space of each region; (2) select a point for each area and solving TSP; (3) improve TSP solution with TPP method. The novelty of Mennell's approach is the concept of running TSP algorithm on skimmed regions. However, the method assumes the centre point of the region is the starter point. This implies a slower convergence rate because optimal via-point is usually located on the boundary.

Elbassoni et al. [36] proposed a method based on Euclidian distance. Firstly, the algorithm sorts regions by size and then it discretises regions by set of points. Starting from the smallest region an inner point is picked up. Next point is selected as the nearest point to the previous one. When all regions are described by inner point a TSP tour is generated.

Alatartsev et al. [45] proposed the CIH method based on Elbassoni and Mennell-like approach (TSP + TPP). Authors proposed an algorithm structured as follow: (1) generate an initial convex hull border tour; (2) insertion of a new region with the nearest centre to the previous tour optimising via-point by rubber-band algorithm (RBA); (3) repeat step 2 up to visiting all regions. Although CIH simultaneously solves via-point allocation and sequence position, it is structured in a sequential manner. Another issue is related to the new region selection; indeed, it is based on minimum distance between region centre and tour. Let's consider two regions with different size, the smallest one presents the nearest

centre, whereas the biggest one presents the nearest boundary. CIH select the smallest one although the nearest is the biggest one.

All these methods have articulated solution to select via-point which are then passed over to classical TSP solver to compute the optimal sequence. This highlights that the leading challenge is related to the via-point selection.

This chapter presents a new enhanced heuristics method based on Euclidian distance to select best via-points in continuous regions. Then, TSP tour construction and TSP tour improvement methods are used. Comparative benchmarking results are then showed to prove the effectiveness of the proposed method.

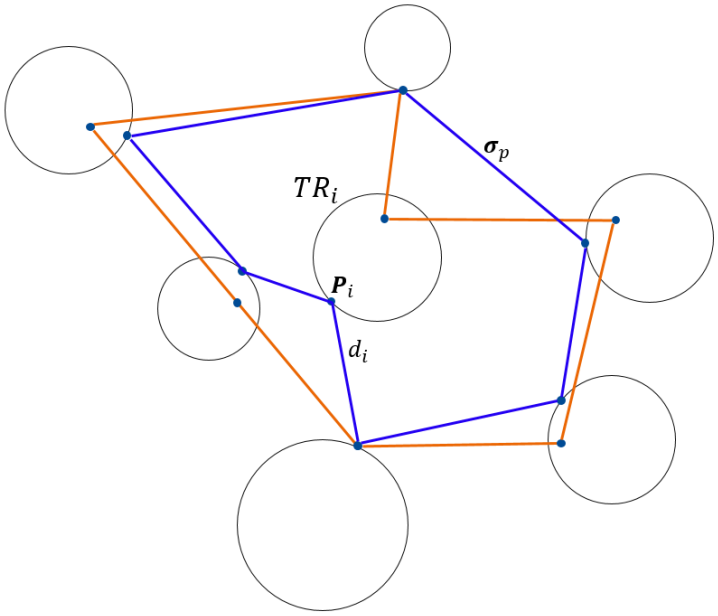

Figure 12 – TSPN formulation. Blue line represents minimum cost tour

4.3.1 Problem formulation

Let tour $\boldsymbol{\sigma}_p = \{\boldsymbol{P}_i\}, i = 1, \dots, n$ be the sequence of via-points, $\boldsymbol{P}_i$, to be optimised within neighbourhoods which are locus of feasible solutions.

Given a set of regions TRs we aim to find the minimum cost tour $\boldsymbol{\sigma}_p$ that visits each region once (Figure 12). The cost associated to the sequence is defined by Euclidean distance; that means, the minimal cost tour represents the shortest path, d_{min}.

$$\forall\, TR_i \in TRs\ \exists\ \boldsymbol{P}_i\ : \boldsymbol{\sigma}_p = \{P_1, \dots P_n\} \Rightarrow d_{min} \qquad with: \quad i \in [1,n]$$

n no. of TR

$$\boldsymbol{P}_i = (x_i, y_i, z_i)$$

$$d_i = \overline{P_i P_{i-1}}$$

$$d_{min} = \sum_{i}^{n} \frac{1}{d_i^2} \tag{2}$$

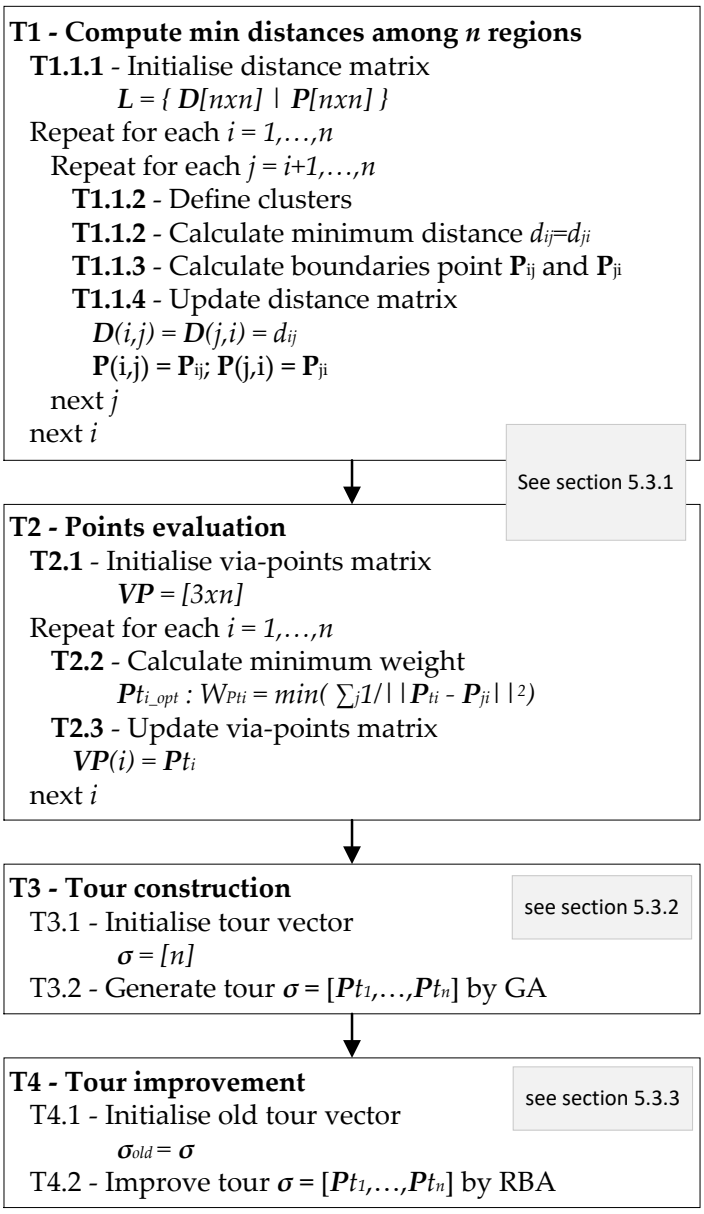

Figure 13 – EH²C flowchart

4.3.2 Proposed Approach

EH^{2}C splits TSPN in four sub-problems: (T1) compute minimum distances among regions; (T2) points evaluation; (T3) tour construction; (T4) tour improvement (see Figure 13).

4.3.2.1 Initial Via-Points Selection (T1 and T2)

EH^{2}C uses Euclidian distance as a key metric for selecting initial points. First of all, minimum distances among clustered regions are computed. For instance, d_{ij} represents the minimum distance between i^{th} and j^{th} region. This distance is associated to two boundary points P_{ij} and P_{ji} located on the two boundaries, respectively.

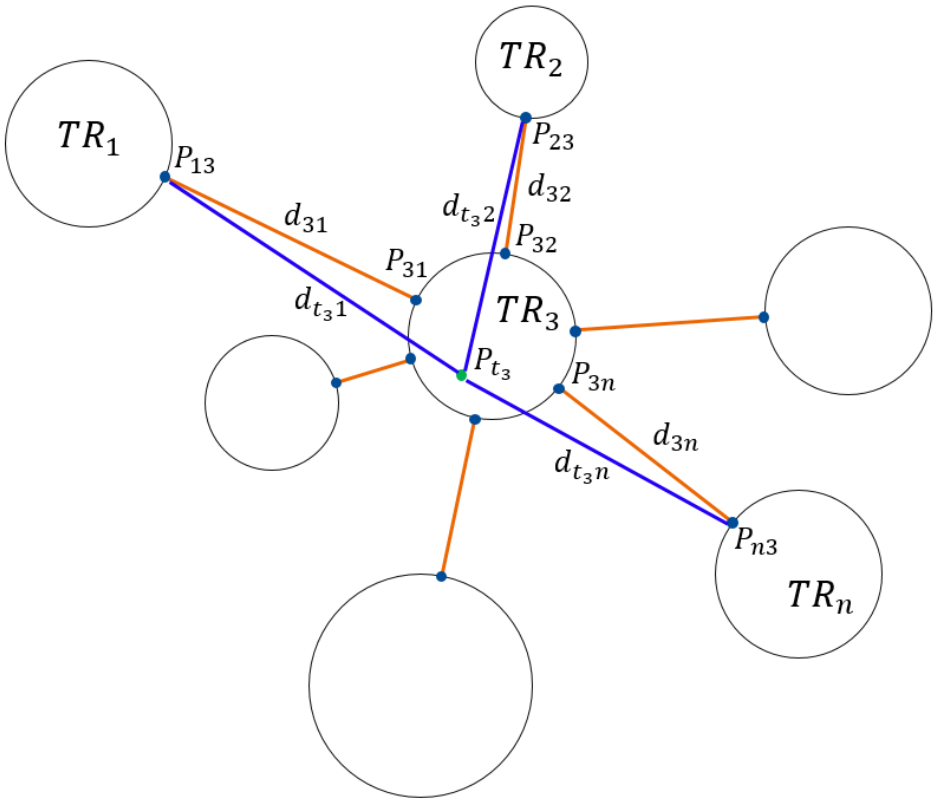

Figure 14 - Initial via-point selection

Assuming that a straight-line represents the minimum route length, the straight-line $\overline{P_{ij}P_{ji}}$ is the best route to move from i^{th} to j^{th} region and vice versa. This assumption translates the minimum energy concept: P_{ij} represents the starting point which corresponds the minimum energy consumption for moving on P_{ji}. Therefore, as P_{ij} is the

best point to move from i^{th} to j^{th} region, the two boundary points will be named as target points.

Considering that a tour consists of multiple regions, multiple boundary points have to be simultaneously computed, that is, tour can be represented as multiple connected straight-line. Referring to Figure 14, for any points P_{t_i} we can define a weight, $W_{P_{t_i}}$:

$$W_{P_{t_i}} = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\left\| P_{t_i} - P_{ji} \right\|^2} \tag{3}$$

For each region, points with minimum weight are selected as via-points. This is formulated as follows:

$$\begin{aligned} &\underset{P_{t_i} \in TR_i}{argmin}(W_{P_{t_i}}) \\ &\forall\, i \in [1, n] \end{aligned} \tag{4}$$

Note is made that the selection of the optimisation algorithm depends on the complexity of the problem. We have used genetic algorithm (GA) because it is flexible enough to be tailored to a wide class of problems, from small to medium large number of tasks. Nevertheless, other optimisation strategies could be implemented in this stage of the methodological work-flow.

Via-points selection is based on minimum energy principle; all via-points represent points with minimum weight. Indeed, P_{t_i} represents the via-point with minimum amount of energy consumption for moving on any P_{ji}.

4.3.2.1.1 Clustering

Regions clustering is based on agglomerative technique of hierarchical and k-means algorithms. K-means aims to partition a set of n elements into k clusters minimising the squared error between the empirical mean of a cluster and the point in the cluster [59]. Hierarchical clustering can be used to generate a partition by specifying a threshold on the similarity (see among others [60], [59]). Hierarchical clustering is often portrayed as the better-quality clustering approach, but is limited because of its quadratic time

complexity. In contrast, K-means and its variants have a time complexity that is linear in the number of documents, but are thought to produce inferior clusters.

We use a K-means algorithm that operates in hierarchical manner for clustering regions which respect to a distance threshold. The method operates on 3 consecutive steps:

(1) Clustering (Step 1). It operates by using the central points of each region. Let $k_{cluster}$ be the number of calculated clusters. Note is made that $k_{cluster}$ is hierarchically incremented until a convergence threshold is reached. The proposed algorithm selects via-points for each region associated to the i^{th} cluster;

(2) Via-point selection (Step 2). It calculates via points by running T2 and T2 steps of the main methodological workflow (see Figure 13); and,

(3) Outermost via-point optimisation (Step 3). The outermost via-point P_{os}^{i} corresponds to the point belonging to the s^{th} region and enclosed by the i^{th} cluster, and is the closest to the neighbourhoods. For example, in Figure 15, the outermost point of the cluster1 is P_{o3}^{1} which belongs to TR_3 and is closest to the cluster 2 and 3.

Having computed the outermost points, the tour is then calculated on the pre-computed points (see T3).

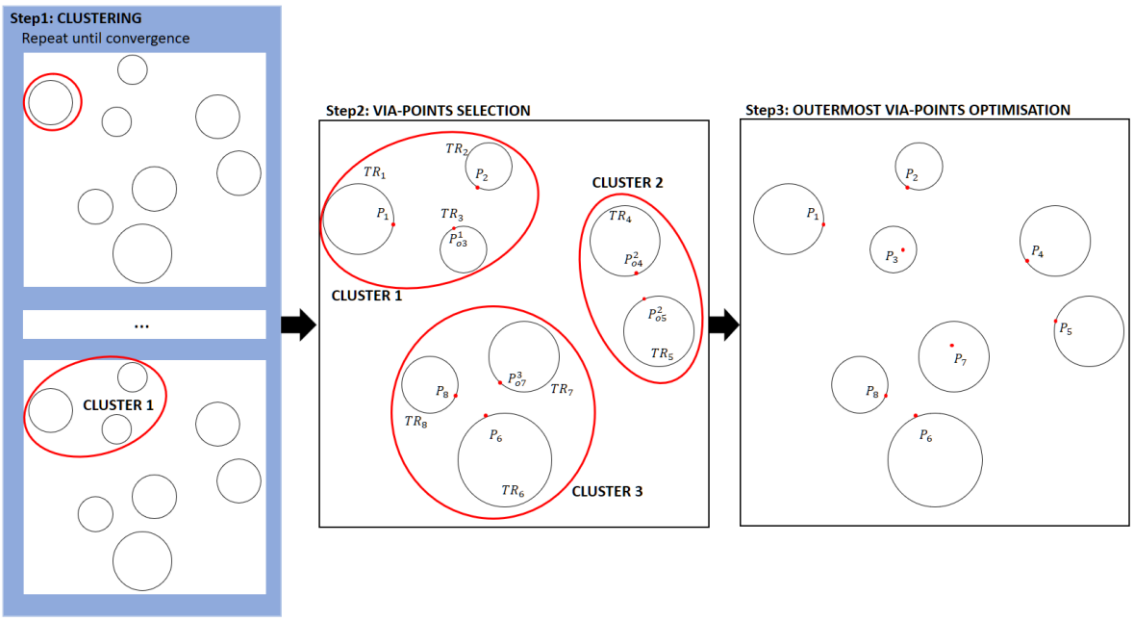

Figure 15 – Hierarchical K-means clustering

4.3.3 Tour Construction (T3)

Having computed via-points tour is constructed using classical TSP solver. We have implemented a robust TSP based on GA. GA has been triggered by initial random population. Tournament selection method has been used to initiate the 2-point cross-over, followed by flip-swap-slide mutation. Fixed number of iterations has been adopted as termination criterion. The effectiveness of the implemented TSP solver has been validated against reference solutions (please refer, for example at [61]).

4.3.4 Tour Improvement (T4)

RBA [62][63] is used to further improve solution. It works on a fixed tour sequence improving via-points location inside each region to reduce tour length. It is based on Euclidian distance and on the assumption that straight-line represents minimum length between two points.

Considering a group of three consecutive via-points, it optimises the middle one: P_i (Figure 16). If the minimum length between P_{i-1} to P_{i+1} is a straight-line, the best position of P_i is on that line. Therefore, the algorithm moves P_i inside TR_i to obtain the minimum distance between P_i and segment $\overline{P_{i-1}P_{i+1}}$.

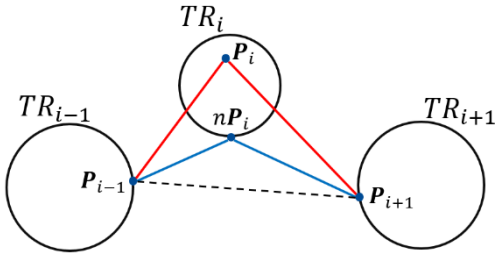

Figure 16 – Point $\boldsymbol{P_i}$ moves into a new position $\boldsymbol{nP_i}$.

Figure 17 shows a 2D example with improvement via-points selection based on RBA.

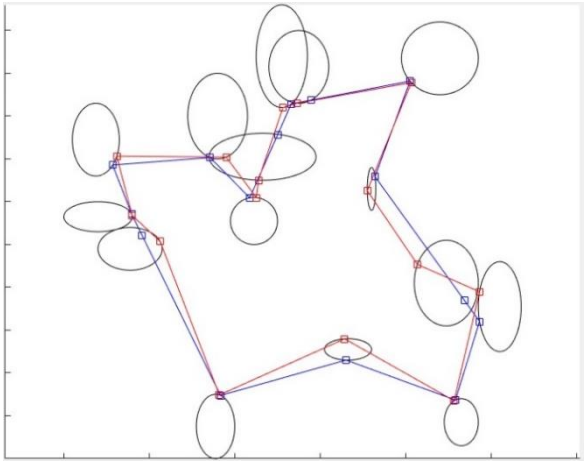

Figure 17 – red squares: initial via-points; red solid-line: initial tour; blue squares: improved via-points; blue solid-line: improved tour.

4.4 Results and Discussion

EH2C has been tested on TSPN benchmarks available in literature. Gentilini et al. [46] provide a set of 64 TSPN test instances formed by ellipsoids and polyhedral in $\mathbb{R}^2$ and $\mathbb{R}^3$. All test instances are available from [65]. Test instances have coded name, for example "tspn2DE15" that means 2D test with 15 ellipses. Besides, they provide the optimal solution for each test.

Using 2D ellipses instances, we compare EH2C with the optimal values and two different algorithms: CIH developed by Alatartsev et al. [45] and BONMIN developed by Gentilini et al. [46]. Results are presented in Table 1. Tour generated by BONMIN and EH2C for instance tspn2DE15_1 are depicted in Figure 18.

Using 3D ellipsoid instances, we compare EH2C with the optimal values and BONMIN. Comparing results are presented in Table 2. Tour generated by EH2C for instance tspn3DE12 is depicted in Figure 19.

Please note that either ellipses or ellipsoids refers to task regions, TRs. Alatartsev et al [45] provide 2D test instances up to 70 ellipses. Tests are available from [66]. Test instances have coded name, for example "30_1_5" that means one of the axis radius stretched from 1 to 5 times in comparison to the other axis radius. Using these instances, we compare EH2C with the optimal values and CIH. Comparing results are presented in Table 3. Diagrams comparing errors and computational times are depict in Figure 20. The proposed algorithm has been implemented in C++ and linked to MatLAB via MEX

interface. Tests have been ran on Intel Core i7-3630QM CPU 2.4GHz with 16GB RAM running Windows 10. Reference computational times was calculated with different machine:

- CIH (Alatartsev et al. [45]) - Intel Core 2 Quad CPU 2.83 GHz and 8 GB RAM running Windows Vista;
- BONMIN (Gentilini et al. [46]) - Dell Precision T7500 with Intel Xeon 3.33 GHz CPU and 12 GB RAM running Fedora 14.

Table 1 - Comparison of EH^{2}C, CIH and BONMIN on TSPN instances from 5 to 16 ellipses

Instance	Optimal	BONMIN		CIH		EH^{2}C	
	value	error(%)	t(s)	error(%)	t(s)	error(%)	t(s)
tspn2DE5_1	191.255	0.00	0.14	0.00	0.00139	1.09e-4	0.29929
tspn2DE5_2	219.307	0.00	0.13	0.00	0.00093	-2.10e-4	0.25558
tspn2DE6_1	202.995	0.00	0.24	0.00	0.00149	3.83e-3	0.25957
tspn2DE6_2	248.860	0.00	0.18	0.00	0.00145	1.96e-4	0.27113
tspn2DE7_1	201.492	0.00	0.3	0.00	0.00646	8.60e-2	0.26886
tspn2DE7_2	239.788	0.00	0.25	0.00	0.00286	8.99e-4	0.26632
tspn2DE8_1	190.243	0.00	0.37	0.28	0.00046	-9.46e-5	0.28076
tspn2DE8_2	229.150	0.00	0.4	0.00	0.00534	0.29	0.29166
tspn2DE9_1	259.290	0.00	0.4	0.00	0.00859	4.23	0.28528
tspn2DE9_2	262.815	0.00	0.41	0.00	0.00704	1.861e-4	0.27524
tspn2DE10_1	225.126	0.00	0.41	0.00	0.00850	0.15	0.28516
tspn2DE10_2	273.192	0.21	0.35	0.00	0.00882	0.21	0.28464
tspn2DE11_1	247.886	0.75	0.63	0.00	0.01221	2.80e-5	0.29400
tspn2DE11_2	258.003	0.00	0.39	0.00	0.01236	6.66e-5	0.30048
tspn2DE12_1	265.858	0.00	0.55	0.00	0.01490	4.13e-5	0.30796
tspn2DE12_2	312.493	0.50	0.86	0.00	0.01916	0.27	0.30329
tspn2DE13_1	278.876	0.00	1.15	0.00	0.02400	7.52e-5	0.33388
tspn2DE13_2	324.271	0.20	0.49	0.00	0.02278	0.21	0.31025
tspn2DE14_1	310.794	0.00	0.95	0.00	0.03766	2.82e-2	0.31652
tspn2DE14_2	270.638	0.56	0.69	0.04	0.02693	4.81	0.31630
tspn2DE15_1	289.716	0.22	1.08	0.00	0.04500	0.20	0.32337
tspn2DE15_2	293.357	0.01	1.20	1.36	0.04731	3.96e-2	0.33464
tspn2DE16_1	369.945	1.09	2.84	6.24	0.04467	4.21	0.33365
tspn2DE16_2	295.130	0.00	1.20	0.00	0.05375	7.53	0.36789

Table 2 - Comparison of EH²C and BONMIN on TSPN instances from 5 to 12 ellipsoids

Instance	Optimal value	BONMIN		EH²C	
		error(%)	t(s)	error(%)	t(s)
tspn23DE5	253.495	0.00	0.20	-1.43e-4	0.29600
tspn3DE6	276.996	0.00	0.27	1.35e-4	0.25854
tspn23DE7	323.689	0.00	0.32	8.50e-5	0.26130
tspn3DE8	296.918	0.00	0.46	0.9225	0.26813
tspn3DE9	312.920	0.0091	0.44	1.7946	0.28457
tspn3DE10	328.627	0.00	0.73	0.3923	0.28876
tspn3DE11	301.307	0.00	0.58	0.0016	0.29771
tspn3DE12	320.575	0.00	1.32	1.3513	0.31169

Table 3 - Comparison of EH²C and CIH on TSPN instances from 20 to 70 ellipses

Instance	Optimal value	CIH		EH²C	
		error(%)	t(s)	error(%)	t(s)
20_1_1	320.720	1.81	0.089	5,35	0.43153
20_1_5	313.497	3.11	0.101	2,04	0.38049
20_1_10	276.793	0.00	0.182	-1,80	0.39265
30_1_1	383.578	1.46	0.363	1,69	0.44931
30_1_5	316.922	0.00	0.443	3,62	0.47585
30_1_10	321.188	0.00	0.654	-3,14	0.45994
40_1_1	421.339	2.41	0.625	-0,61	0.53795
40_1_5	368.802	0.00	1.140	4,78	0.61665
40_1_10	312.353	0.75	1.211	11,28	0.57299
50_1_1	438.182	4.27	1.595	-0,27	0.64281
50_1_5	457.114	2.12	1.904	0,60	0.71835
50_1_10	397.472	3.34	2.182	15,66	0.70082
60_1_1	563.603	7.99	2.355	6,87	0.78522
60_1_5	563.438	0.38	2.320	5,89	0.79706
60_1_10	499.973	3.60	2.621	5,71	0.78940
70_1_1	622.098	3.39	3.326	7,23	0.93148
70_1_5	587.004	3.43	3.921	5,23	0.95362
70_1_10	509.905	0.02	4.713	4,56	0.97601

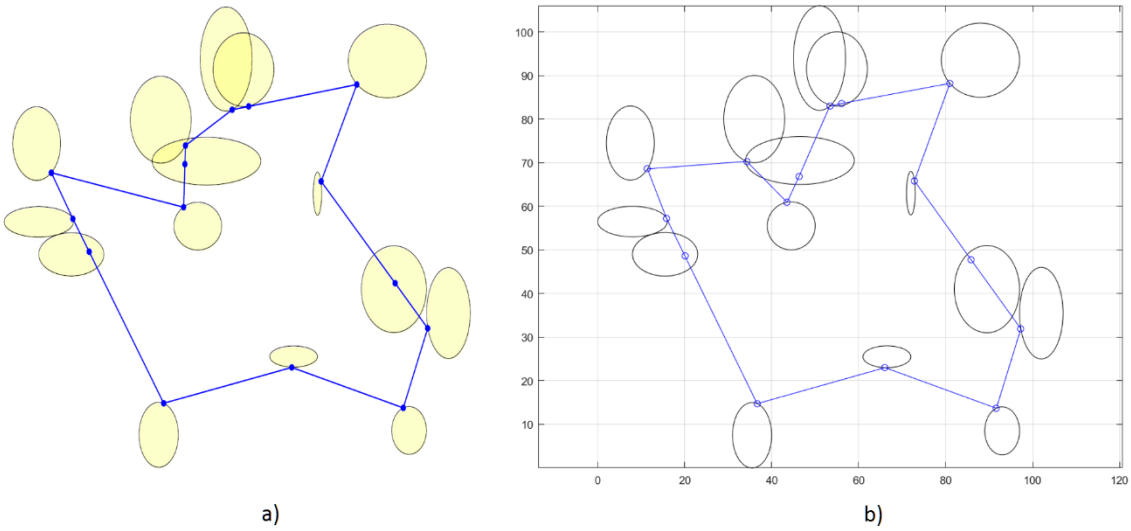

Figure 18 - tour generated for instance tspn2D15_1: a) BONMIN solution [46]; b) EH^{2}C solution

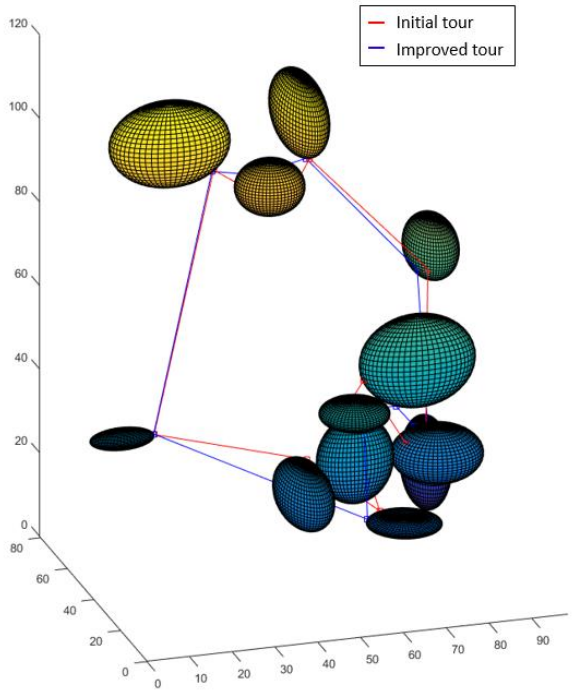

Figure 19 - Instance tspn3DE12 solved by EH^{2}C

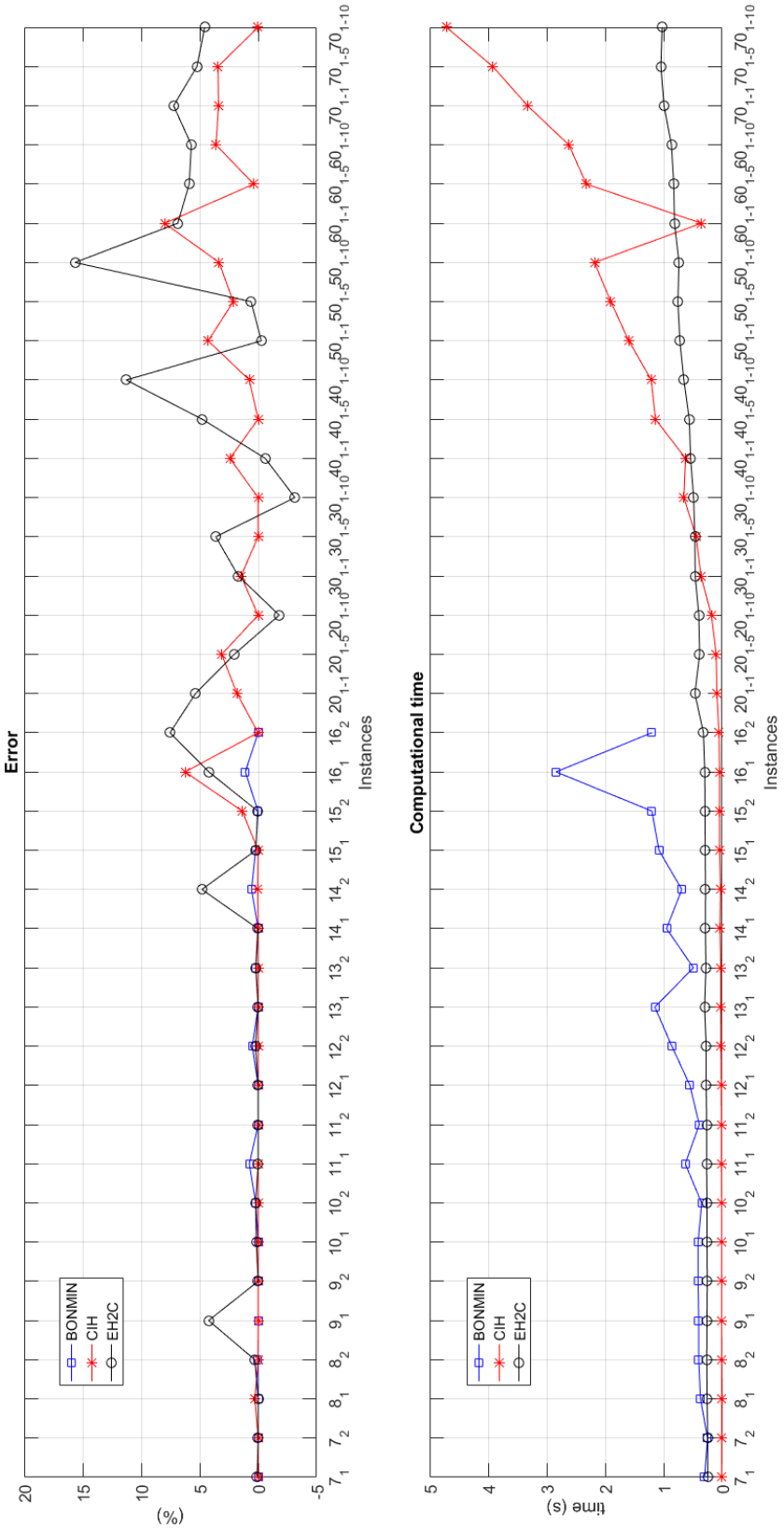

Figure 20 – Comparing errors and computation times among BONMIN, CIH and EH2C

Results show that computation time for all methods increases as the number of tasks, $\boldsymbol{n}$, increases. This trend was expected because the complexity for solving test instances increases as $\boldsymbol{n}$ increases. Complexity of computational process can be assessed by comparing computation time for tspn2DE16 (Table 1) and computation time for tspn2D5 (calculated as average between instance "…_1" and "…_2"). BONMIN increases by 15 times in computation time, CIH by 42 and EH2C by only 1.2. This results translate saying that the computation time of CIH is strongly coupled to $\boldsymbol{n}$, although it is the fastest method up to 30 ellipses. BONMIN is the slowest one but the most accurate and its complexity increases less than CIH. EH2C appears to have the most effective computation process. Indeed, the number of neighbourhoods affects slightly the computation time. Furthermore, EH2C is only slightly coupled to $\boldsymbol{n}$. This aspect is strengthened by test instances up to 70 ellipses. Comparing computation time for 70 and 20 ellipses (calculated as average among "…_1_1", "…_1_5" and "…_1_10"), EH2C only increases by 2 times in computation time while CIH does by 32. To better understand how computation time increases as $\boldsymbol{n}$ increases we can compare instances "70_..." (Table 3) with "tspn2D5_..." (Table 1). Results show that CIH increases by 3437 times in computation time; whereas EH^2C increases only by 3.3.

Table 4 shows fractions of the time needed for solving 2D instances. Data show that, unsurprisingly, the computation time for all steps increases as n increases. Unexpected results are related to step computation time of T3; indeed, it almost takes the majority of the time (90%) with few instances up to 50% with 70 ellipses. This is in contrast with all other methods where time for initial tour construction is the smallest and however smaller than time for tour improvement [46]. Other important result is related to step T1: the computation time rate increases more than others up to 25% for 70 ellipses. Therefore, we can conclude that step T1 predominate steps T2 to T4, when the number of instances gets relatively large.

Same patterns for T1, T2, T3 and T4 are highlighted by 3D instances (Table 5). Besides, comparing the time needed for solving ellipses and ellipsoids there are no differences; this means that EH^2C is not affected by neighbourhoods' geometry. This is in contrast with others methods. For example, Gentilini et al. [46] claim that 3D instances are harder to solve than 2D ones, given the same number n of neighbourhoods. Besides,

authors state that dimensional factor affects computation time too, i.e. larger neighbourhoods are harder to solve than instances with smaller ones when number n is fixed. However, there are instances for which the opposite is true; therefore, other aspects not considered in the analysis may also influence computation time.

Table 4 – Fractions of time for solving 2D instances by EH^{2}C

Instance	Step T1 (s) distances	Step T2 (s) via-points	Step T3 (s) construction	Step T4 (s) improvement	TOT (s)
tspn2DE5_1	0,00609	0,03988	0,23664	0,01668	0,29929
tspn2DE5_2	0,00210	0,00994	0,23148	0,01206	0,25558
tspn2DE6_1	0,00223	0,00731	0,23542	0,01461	0,25957
tspn2DE6_2	0,00205	0,00677	0,24917	0,01313	0,27113
tspn2DE7_1	0,00283	0,01098	0,24140	0,01365	0,26886
tspn2DE7_2	0,00267	0,00895	0,24118	0,01352	0,26632
tspn2DE8_1	0,00386	0,01271	0,24670	0,01750	0,28076
tspn2DE8_2	0,00385	0,01659	0,25141	0,01981	0,29166
tspn2DE9_1	0,00548	0,00839	0,25285	0,01857	0,28528
tspn2DE9_2	0,00404	0,00831	0,24953	0,01336	0,27524
tspn2DE10_1	0,00492	0,00866	0,25630	0,01528	0,28516
tspn2DE10_2	0,00507	0,00894	0,25661	0,01402	0,28464
tspn2DE11_1	0,00641	0,01040	0,26153	0,01566	0,29400
tspn2DE11_2	0,00629	0,00978	0,26958	0,01484	0,30048
tspn2DE12_1	0,00729	0,01023	0,27236	0,01808	0,30796
tspn2DE12_2	0,00841	0,01124	0,26608	0,01756	0,30329
tspn2DE13_1	0,00872	0,01162	0,26774	0,04581	0,33388
tspn2DE13_2	0,00928	0,01263	0,26924	0,01909	0,31025
tspn2DE14_1	0,01050	0,01347	0,27155	0,02100	0,31652
tspn2DE14_2	0,01050	0,01387	0,27209	0,01986	0,31630
tspn2DE15_1	0,01153	0,01319	0,27878	0,01986	0,32337
tspn2DE15_2	0,01311	0,01922	0,28044	0,02187	0,33464
tspn2DE16_1	0,01327	0,01898	0,27810	0,02330	0,33365
tspn2DE16_2	0,01696	0,01902	0,30130	0,03061	0,36789
20_1_1	0,02858	0,07711	0,28053	0,04530	0,43153
20_1_5	0,02294	0,02896	0,28155	0,04704	0,38049
20_1_10	0,02281	0,03983	0,27633	0,05368	0,39265
30_1_1	0,04372	0,04386	0,31158	0,05014	0,44931
30_1_5	0,04556	0,03306	0,31948	0,07775	0,47585
30_1_10	0,04539	0,04066	0,31944	0,05444	0,45994
40_1_1	0,07778	0,04241	0,36957	0,04819	0,53795
40_1_5	0,08987	0,06558	0,37158	0,08963	0,61665
40_1_10	0,08079	0,04093	0,36406	0,08721	0,57299
50_1_1	0,12448	0,05324	0,40909	0,05600	0,64281

50_1_5	0,12494	0,07239	0,41832	0,10269	0,71835
50_1_10	0,12981	0,06171	0,41503	0,09427	0,70082
60_1_1	0,17937	0,08087	0,46062	0,06436	0,78522
60_1_5	0,18394	0,08000	0,46578	0,06734	0,79706
60_1_10	0,18260	0,07427	0,46192	0,07062	0,78940
70_1_1	0,24672	0,09154	0,51575	0,07747	0,93148
70_1_5	0,24873	0,09347	0,51123	0,10018	0,95362
70_1_10	0,24713	0,09094	0,51197	0,12596	0,97601

Table 5 - Fractions of time for solving 3D instances by EH^{2}C

Instance	Step T1 (s) distances	Step T2 (s) via-points	Step T3 (s) construction	Step T4 (s) improvement	TOT (s)
tspn23DE5	0,00994	0,04979	0,22008	0,01620	0,29600
tspn3DE6	0,00717	0,01105	0,22317	0,01715	0,25854
tspn23DE7	0,01013	0,00885	0,22777	0,01455	0,26130
tspn3DE8	0,01148	0,01048	0,22953	0,01664	0,26813
tspn3DE9	0,01472	0,01489	0,23800	0,01695	0,28457
tspn3DE10	0,01959	0,01142	0,23857	0,01918	0,28876
tspn3DE11	0,02248	0,01164	0,24505	0,01855	0,29771
tspn3DE12	0,02787	0,01444	0,24759	0,02179	0,31169

We have generated a quantitative index to predict model complexity, as represented equations (5) and (6). We consider only ellipse instances neglecting geometric and dimensional factors.

$$\text{CIH:} \quad t = 0.001n^2 - 0.02n + 0.1122 \tag{5}$$

$$\text{EH}^2\text{C:} \quad t = 0.0118n + 0.1448 \tag{6}$$

Computation time and forecasted time by equations (3) and (4) are represented in Figure 21. Time needed for solving instances by CIH follows a 2nd degree polynomial; whereas EH^{2}C is approximated with a linear polynomial.

Accuracy is comparable for all methods up to 15 ellipses. Then, error has fluctuation for both CIH and EH^{2}C and there is no clear pattern. EH^{2}C fluctuations are likely related to hierarchical clustering but, we have not found a clear relation yet. EH^{2}C finds new optimal values compared to the well-known values in literature: for instances 20_1_10,

30_1_10, 40_1_1 and 50_1_1 equal to 271.811, 311.103, 418.769 and 436.999 respectively.

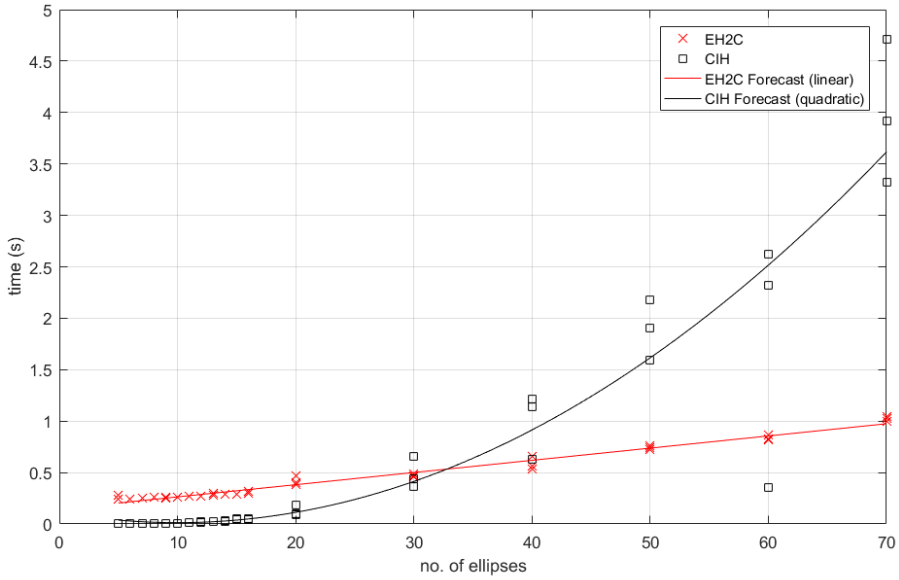

Figure 21 - Polynomial fitting functions

4.5 Summary and Remarks

We have proposed a new method - EH^{2}C – for solving the TSPN problem. It decomposes TSPN into four sub-problems: (T1) find minimum distance among regions; (T2) select best initial via-points for each region; (T3) construct initial tour; (T4) improve tour by modifying via-points.

EH^{2}C was evaluated on three different test instances:

- 2D space with small numbers of ellipses comparing with two different methods;
- 2D space with medium numbers of ellipses comparing with the best existing method;
- 3D space with small numbers of ellipsoids comparing with the best existing method.

Results with small numbers of ellipses n showed that EH^{2}C is able to solve these instances close to the optimum value and for medium n it finds even better solutions compared to the best know solutions. EH^{2}C is the fastest method with medium n and

appears to be the best method in terms of complexity of computational process as it is slightly affected by n.

Benchmarking studies have highlighted that the complexity of the TSPN solution mostly depends on the following properties of the regions: (1) number; (2) topology; (3) size. For instance, BONMIN suffers when solving 3D geometry because of the topological complexity to compute the optimal tour. Instead, the proposed EH^{2}C method appears to be insensitive to topology. Further investigation is needed to better understand the effect of size.

5 AUGMENTED EH^2C FOR TASK SEQUENCING OPTIMISATION

This chapter presents the proposed methodology to solve the robotic task sequencing problem using A-EH^{2}C. The proposed method optimises both sequence and via-poses by augmenting T-space with robot attributes. We have applied A-EH^{2}C for solving task sequencing of an inspection robots with optical camera system.

5.1 Augmented EH^{2}C

We have augmented EH^{2}C algorithm for solving robotic task sequencing problem. It is based on integration of multiple attributes to identify optimal via-poses.

As in T-space there are no robot information, we have defined an index for each attribute in order to evaluate the attribute impact on sequence of tasks.

A-EH^{2}C flowchart is depicted in Figure 22. For each position $\boldsymbol{P}_{t_i}$ within a task region, TR, Euclidean distance attribute W_d is computed (see also Equation (3) in Chapter 4). Then, a set of orientation $\boldsymbol{Or}$ is sampled, to generate a set of poses $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_\mathrm{i}, \dots, \boldsymbol{\lambda}_\mathrm{m}\}$, where $\boldsymbol{\lambda s}$ have same position $\boldsymbol{P}$ and different orientations $\boldsymbol{Ors}$.

Three robot attributes are calculated for each generated pose $\boldsymbol{\lambda}$:

(1) pose accuracy's index λA;

(2) pose reachability's index λR; and,

(3) pose collision's index λC.

Poses eligibility is calculated for each task region TR and sequence of tasks is generated via elected $\boldsymbol{\Lambda}$ which has same position $\boldsymbol{P}$ for all poses $\boldsymbol{\lambda s}$.

In this way, although there is no information on path planning, we can generate an optimal sequence which corresponds to the best *feasible* sequence.

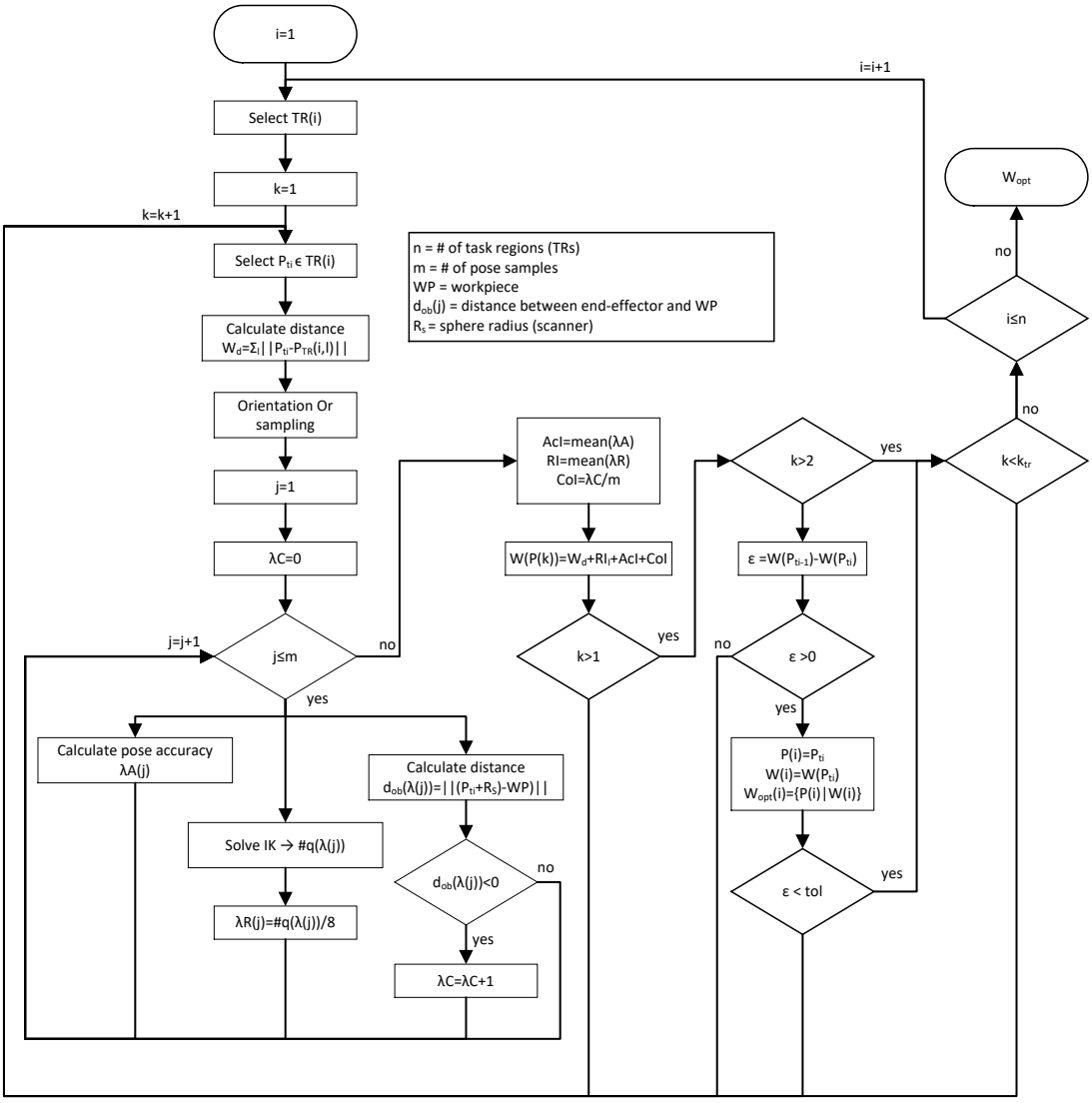

Figure 22 - Flowchart of the A-EH^{2}C

5.1.1 Accuracy Index ($\boldsymbol{AcI}$)

Accuracy index AcI aims to evaluate the quality of task execution. It is calculated as average of all pose accuracy indices λAs within set of poses $\boldsymbol{\Lambda}$.

$$AcI = mean(\lambda As) \tag{7}$$

5.1.2 Reachability Index ($\boldsymbol{RI}$)

Reachability index RI aims to evaluate the feasibility degree of the poses. Solving inverse kinematics for each pose $\boldsymbol{\lambda}$ of the set $\boldsymbol{\Lambda}$, a pose reachability λR is calculated as number of solutions by admissible solutions.

$$\lambda R = \frac{no.of\ solutions}{no.\ of\ admissible\ solutions} \tag{8}$$

RI is calculated as average of all pose reachability indices λRs within set of poses $\boldsymbol{\Lambda}$.

$$RI = mean(\lambda Rs) \tag{9}$$

5.1.3 Collision Index ($\boldsymbol{CoI}$)

Collision index aims to evaluate the collision tendency of a pose. If collision exist, count collision, not count. For each pose $\boldsymbol{\lambda}$, pose collision index λC is 1 if collision exist, otherwise 0.

$$\lambda C = \begin{cases} 1\ \ collision\ true \\ 0\ collision\ false \end{cases} \tag{10}$$

CoI is calculated as average of all pose collision indices λCs within set of poses $\boldsymbol{\Lambda}$.

$$CoI = \lambda Cs/m \tag{11}$$

5.2 Results and Discussion

The proposed methodology has been applied using a robotic metrology 3D scanner measuring the right-front door of automotive SUV (Figure 23).

Door is one of the key element in automotive industries and its measurement is very important to reduce defect such as gap and flush during the assembly process between doors and bodies.

Gap and flush measurement is commonly carried out to inspect fit and alignment between two surfaces, for example, as said before, the gap between car door and body panels.

Figure 23 - Right-front door of automotive SUV door [67]

Each component may be manufactured by different suppliers, potentially in different locations, using many wide and varied processes that eventually all have to fit together to make one product. This means that fit and quality control is vital, gap and flush measurement is highly beneficial. If fit and finish is out of specification it not only affects the aesthetics of the product but also the performance, efficiency and risk of failure.

For this reason, is crucial to be able to measure a door, in short time and with a high accuracy. Figure 24 shows the robot cell installed at University of Warwick, WMG.

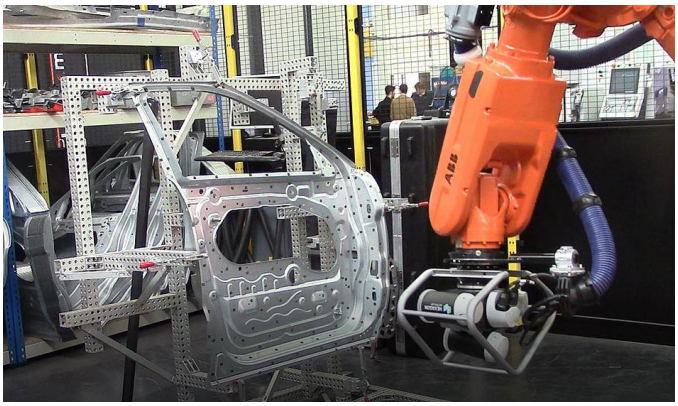

Figure 24 - Robot cell installed at WMG

5.2.1 System Description

The metrology system entails (1) Hexagon Metrology WLS400A (Figure 25) as end-effector on a 6-axis ABB IRB 6620 robot.

Figure 25 - Hexagon Metrology WLS400A

WLS400A is a white light scanner measuring system equipped with 3 x 4.0 megapixel digital cameras. It has a field of view equal to 500 x 500 mm, 230 mm as depth of field and an optimal working distance of 780 mm.

IRB 6620 is a six axes robot. It has a position repeatability of 0.03 mm and his working data are showed in Table 6.

Table 6 - Robot technical data

Axis	Working range	Max. speed [°/s]
1. Rotation	+170° to -170°	100
2. Arm	+140° to -65°	90
3. Arm	+70° to -180°	90
4. Wrist	+300° to -300°	150
5. Bend	+130 to -130°	120
6. Turn	+300 to -300°	190

5.2.2 Task Planning

Task region is the locus of feasible points, which can be visited the end-effector (WLS400A) to perform that task. For a camera vision system, the task corresponds to feasible measurement volume, i.e. the locus of feasible capture points P_C.

Task region is defined by: optimal working distance ($r_{opt} = 780\ mm$); depth of field ($r_{max} - r_{min} = 230\ mm$); minimum reflection feasible angle ($\phi_{min} = 15°$); maximum reflection feasible angle ($\phi_{max} = 35°$).

Region is built assuming focal point P_f at the centre of field of view (Figure 26). r_{opt} generates, around P_f, a sphere surface that represent the locus of optimal capture points. Depth of field implies a variable capture distance within range $[r_{min}, r_{max}]$. This working range distance generates an offset respect to sphere surface. Normal of field of view represents z-axis in the point of focus reference. Feasible angles ϕ_{max}, evaluated respect to z-axis, implies a volume reduction to a sphere calotte. Considering rotation limits of joint 6 (θ_{6min} and θ_{6max}) we overall define task region as depicted in Figure 26.

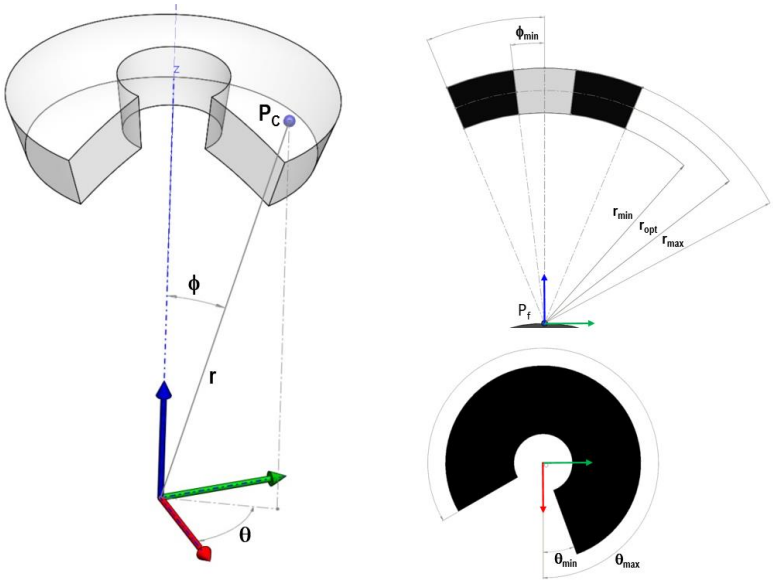

Figure 26 - Task region definition for robotic optical scanner WLS400A

5.2.3 Task Sequencing

Firstly, we have characterised all robot attributes based on the described optical vision system.

5.2.3.1 Pose Reachability Index

ABB IRB 6620 is a 6 axes robot; therefore, it has 6 DoF. Such a robot admits up to 8 solutions; which are the number of admissible solutions.

$$RI = mean\left(\frac{no.of\ solutions}{8}\right)$$

5.2.3.2 Accuracy Index

Accuracy of capture point P_c of the Hexagon WLS400A is related to image capture problem. Accuracy is affected by three main problem: light reflection; material properties of the workpiece and optics.

Light reflection

When we take a photo, is important to illuminate the object so that every point of it is reached by the same amount of light. If the surface isn't flat, however, is possible that some point of the surface may be in the shade. To overcome this problem the position of the light is crucial (Figure 27).

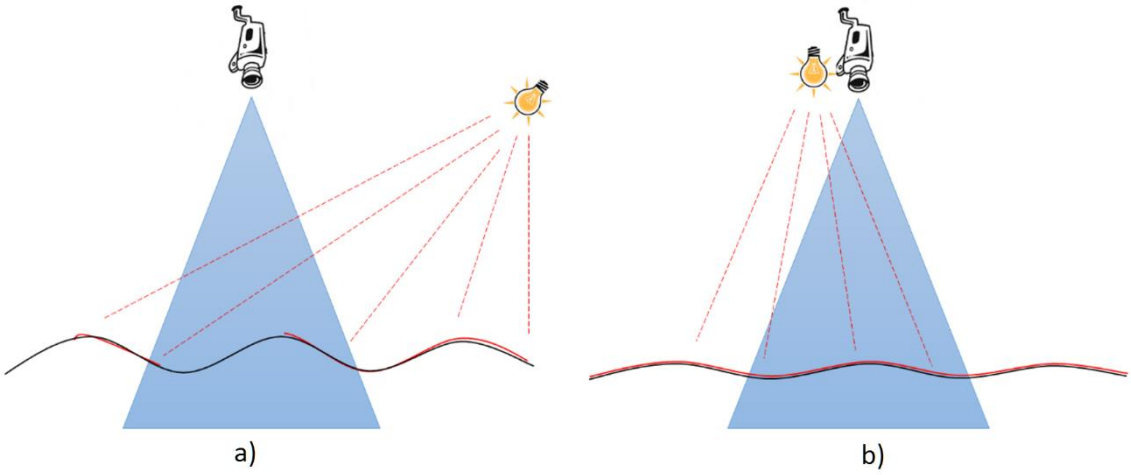

Figure 27 - a) side lighting; b) front lighting

The two figures show with the blue light triangle ( ) the Field of view of the camera in the light direction and with the red waves ( ) the surface reached by the

light. It is clear from Figure 27-a that a side lighting creates more shadows especially if the surface has a high roughness. Figure 27-b Figure 23shows the best configuration with the light source and the camera on the same direction and normal to the surface. In this case, all the point in the Field of View (FoV) of camera are illuminated homogeneously.

Material properties

There are two properties that can influence how light interact with an object; roughness and colour. In Figure 28 is shown in red the specular reflection and in yellow the diffuse reflection (scatter). In the former, the camera is placed normal to the surface in the specular beam, where most likely, the photo will be overexposed. In the latter Figure 28-b is shown what happens if the camera is placed with a higher angle. The lens is far from both beams and the photo will appear darker, underexposed.

For the reasons mentioned above, the best predicted position is within a range between the specular and great angles Figure 29.

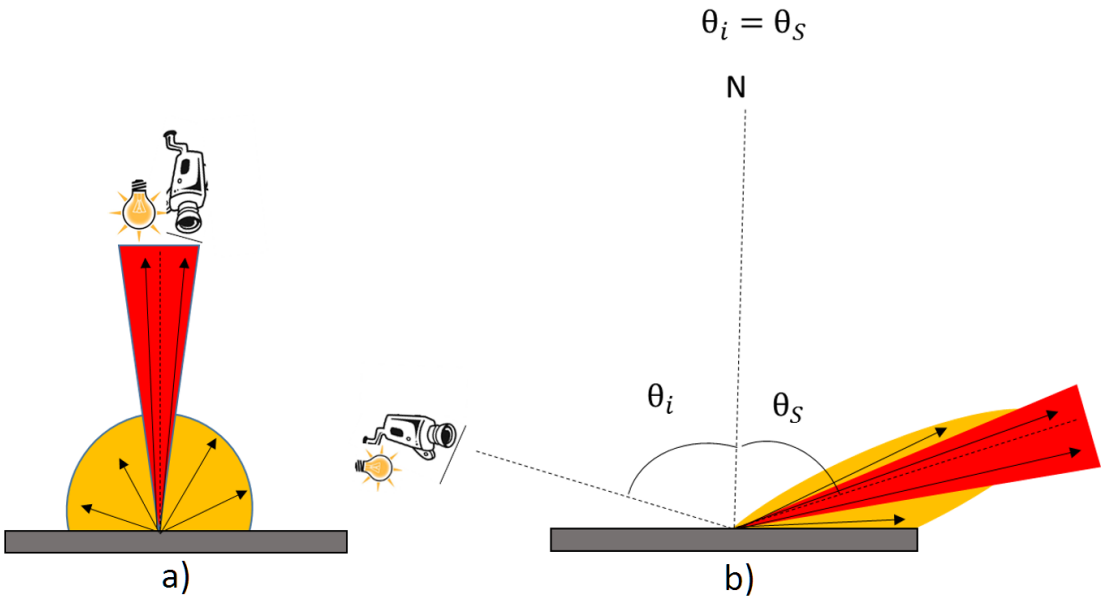

Figure 28 - a) overexposed; b) underexposed

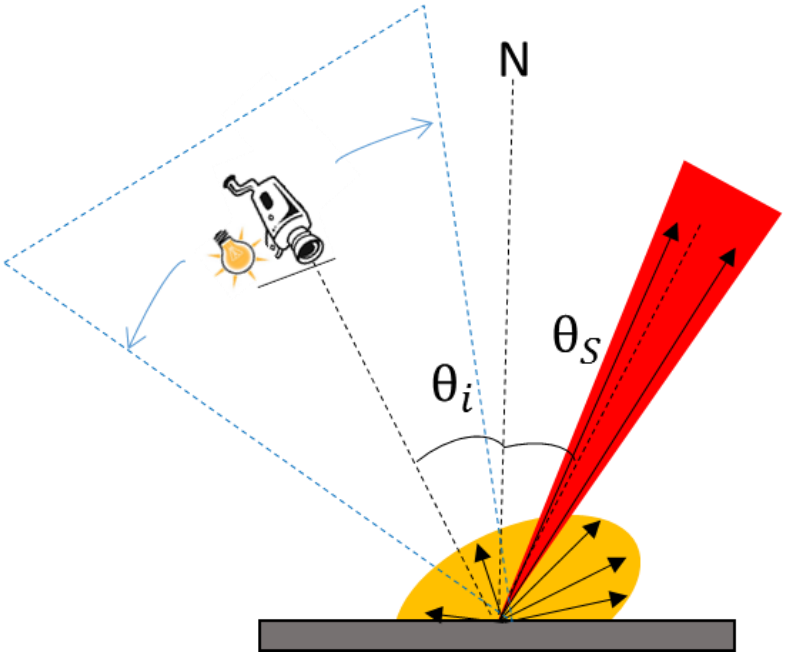

Figure 29 - Optimal expected position

Optics

With the regulation of the exposure in an optical system we can decide the amount of light that reaches the sensor. It means that the system can move closer or farther from the point kipping an acceptable quality image as shown in Figure 30.

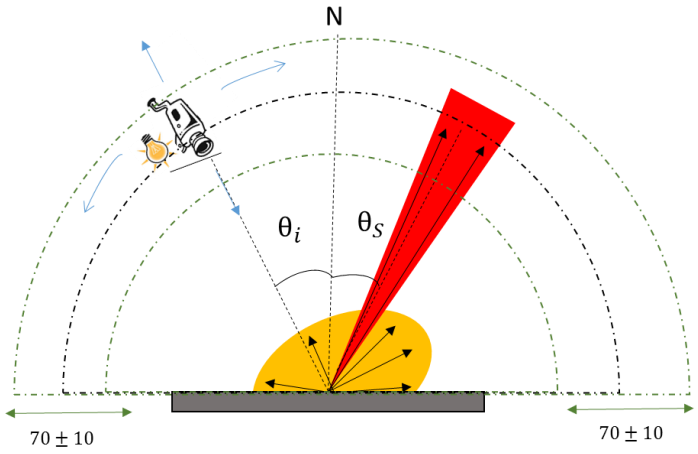

Figure 30 - Exposure

Moving close or further the image quality remain acceptable however there are some drawbacks in both cases Figure 31

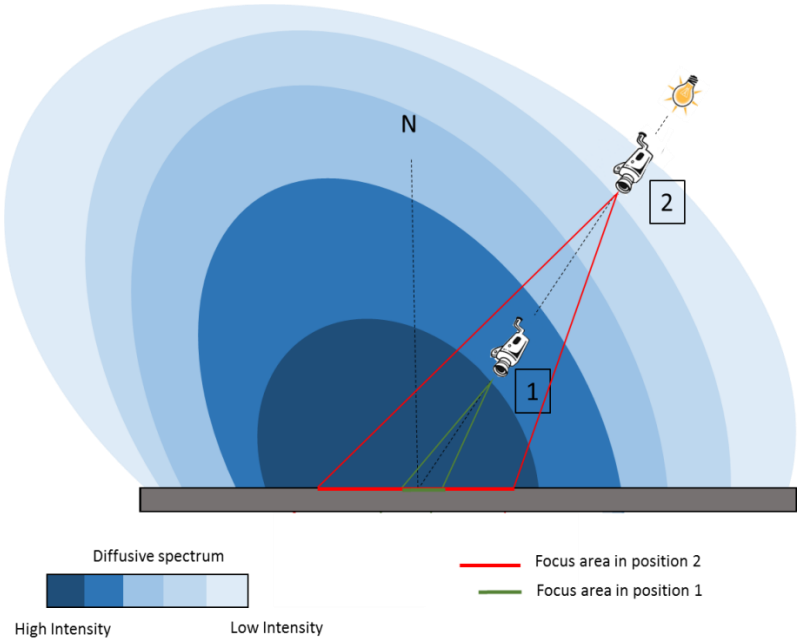

Figure 31 - Quality difference moving from 1 to 2

Moving closer to the point in position 1 higher quantity of light reaches the camera. To ensure a good quality of the photo we should reduce aperture, decrease ISO and increase shutter speed. Moreover, we will focus on a smaller area so less point will be measured. On the other side, moving to position 2 we will focus on a bigger area but since less quantity of light arrives to the sensor we should increase aperture, set higher ISO and change shutter speed. All this changes leads to a lower quality of the image even though more points are measured in once.

CI calculation

For evaluating capture quality we have used the coverage index (CI) which is defined as ratio between valuate area covered by point cloud and nominal area of the geometry. It can assume values between zero and one.

$$0 < Cover\ Index = \frac{Valuate\ Area}{Nominal\ Area} < 1$$

CI is calculated respect incident angle ϕ using a mapping function of WLS400A (Figure 32) developed at WMG (University of Warwick).

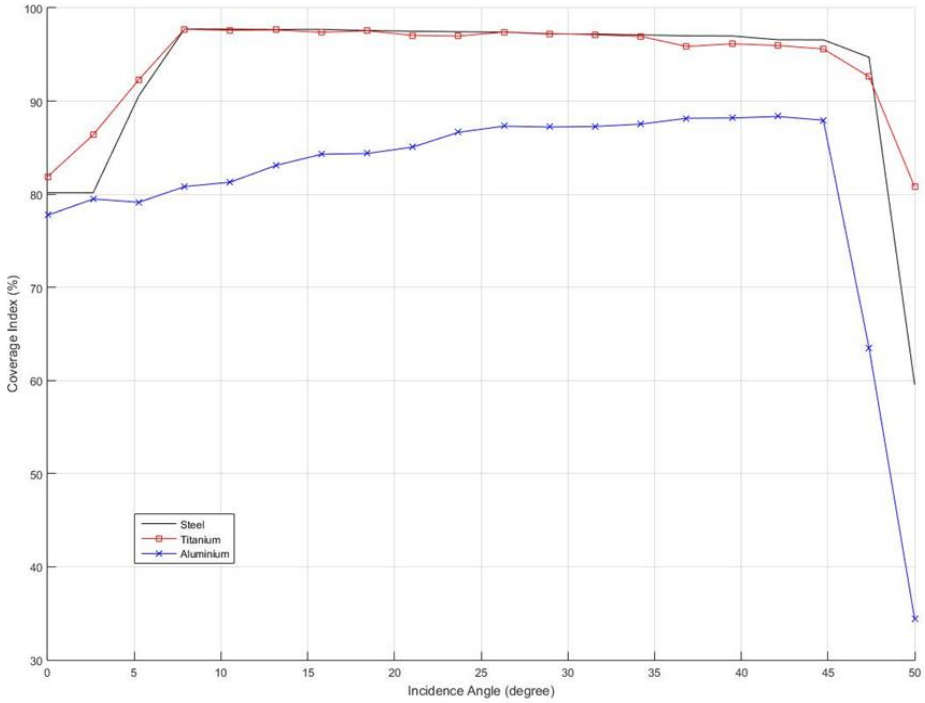

Figure 32 - CI map for WLS400A

Discretising turn angle θ_6 in m angle samples. For a given camera position, we calculate CI as average of CI_θ that is calculated as average of CI_{P_c} between the camera and the sampled points in the FoV (Figure 33):

$$CI = \frac{\sum_i^m CI_{\theta i}}{m} \qquad CI_{\theta_i} = \frac{\sum_j^k CI_{P_{Cj}}}{k}$$

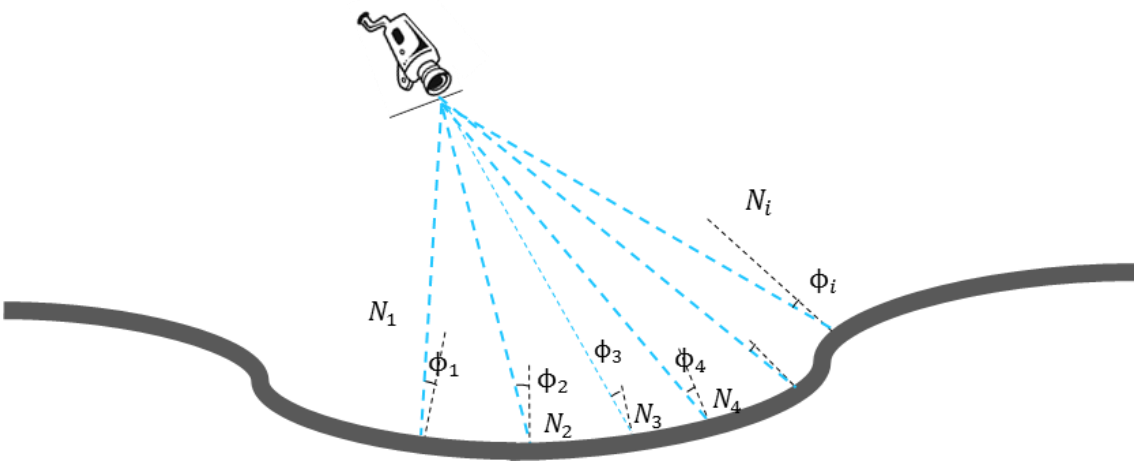

Figure 33 - Sampling points within FoV and related incident angle evaluation

5.2.3.3 Collision Index

We evaluate collision as intersection between card door and a spherical envelop of the scanner. WLS400A scanner is bounded by a structure of protection (Figure 24). Then, the spherical envelope of the scanner is obtained considering as centre of the sphere the mean point of the cage and as radius the biggest distance from a point of the scanner to this mean point (Figure 34).

Collision is defined as binary state: if collision exist 1 (see Figure 35) otherwise 0.

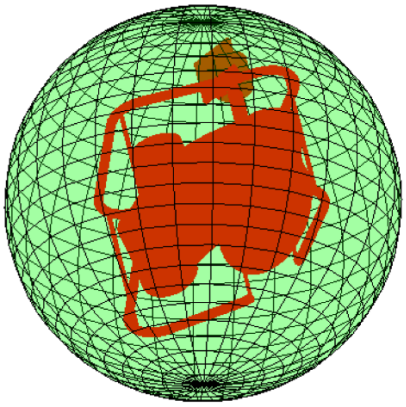

Figure 34 - Scanner envelope

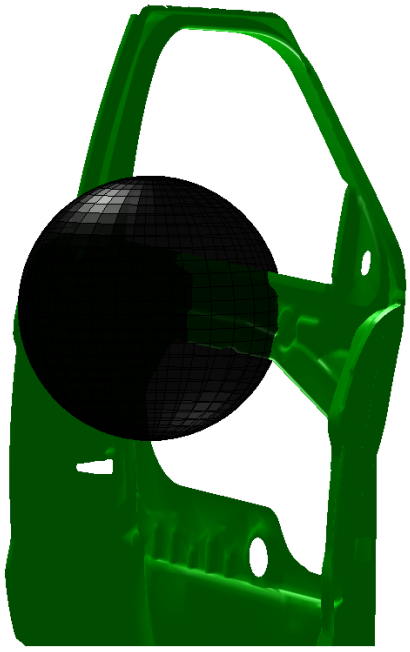

Figure 35 - Collision between scanner envelope and workpiece

5.2.3.4 Case-in-point

Robot cell has been developed in Matlab. The proposed algorithm has been implemented in C++ and linked to MatLAB via MEX interface.

In literature, there is no benchmarks for task sequencing problem, this impeded us to compare our approach with existing ones.

In order to assess the benefits of a such multi-attribute approach, we have compared two task sequencing solutions generated after an iteration by using only one attribute (distance) and all attributes (distance, accuracy, reachability and collision) respectively. In the first case, as depict in Figure 36, obtained via-poses present collisions with the workpiece. Therefore, further iterations are needed to generate a feasible solution and more computation time are required. In the second case, using all attributes, no more actions are needed; indeed, as show in Figure 37, the generated solution is feasible because no collisions occur. Besides, obtained via-points are the optimum ones in terms of measurement quality, robot reachability and path length.

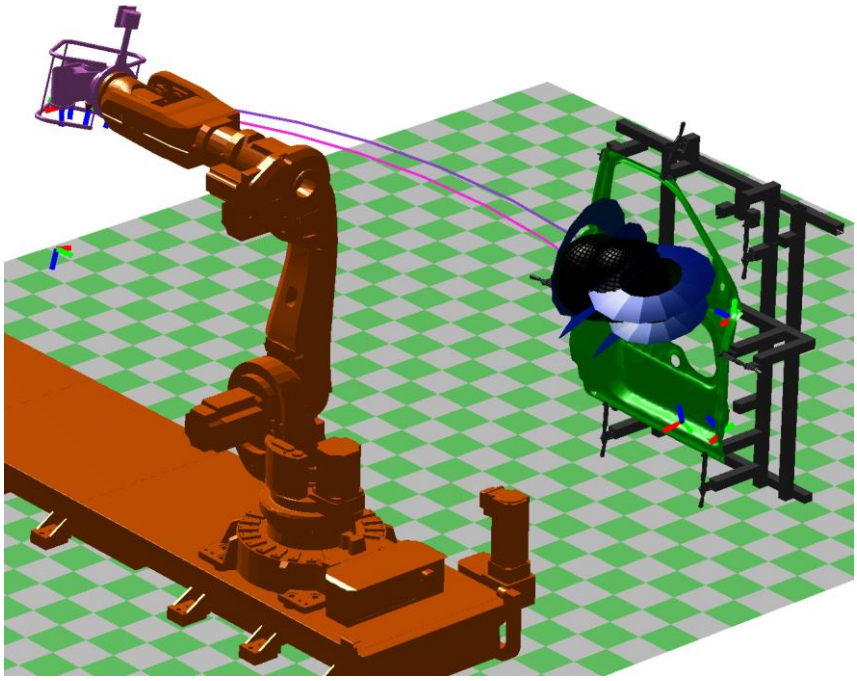

Figure 36 -Robot cell representation in Matlab environment: task sequencing solved with distance attribute

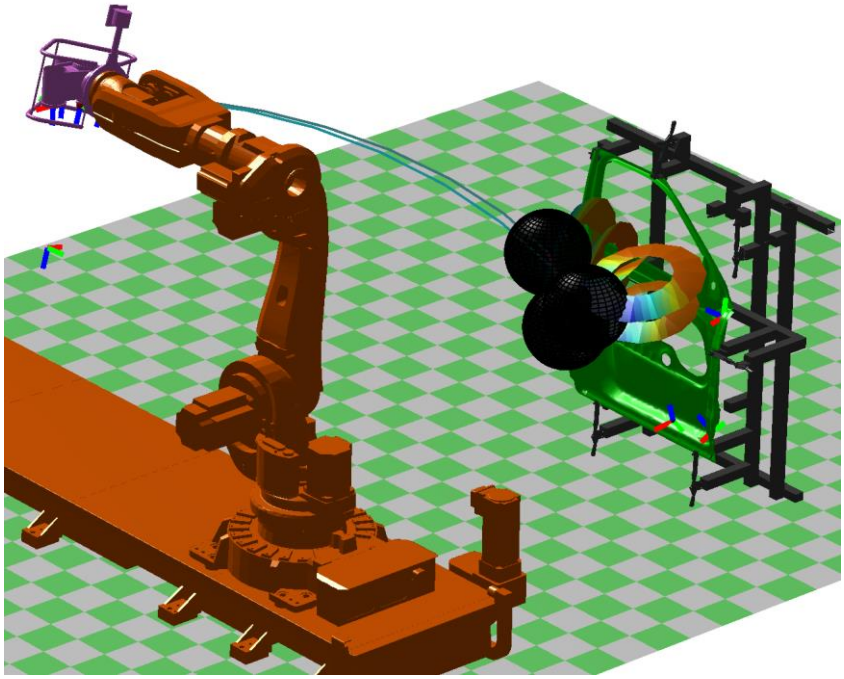

Figure 37 - Task sequencing solved with all attributes: distance, accuracy, reachability and collision

6 CONCLUSIONS AND FURTHER WORKS

This chapter briefly summarises the present research and points out the main contributions. Besides, it discusses further works in robotic task sequencing.

6.1 Conclusions

This dissertation addresses the technical problem of finding near-optimal task sequencing for robotic systems, with the ultimate goal of minimising computational time to enable dynamic robot programming in the case of multiple and coupled tasks' attributes. The dissertation introduces the concept of multi-attribute task sequencing, as a paradigm to solve coupled and hierarchical robotic tasks. Task sequencing problem is modelled using the Travelling Salesman Problem with Neighbourhoods (TSPN) approach.

The dissertation has proposed a new method - Enhanced Heuristic with Hierarchical Clustering (EH^2C) – for solving the TSPN problem using Euclidean distance. It decomposes TSPN into four sub-problems: (T1) find minimum distance among regions; (T2) select best initial via-points for each region; (T3) construct initial tour; (T4) improve tour by modifying via-points. EH^2C was evaluated on three different benchmarks. Results show that EH^2C is able to find near-optimal values, and in some cases the computed solution is even better than the best-known solutions, and faster than well-established heuristic methods.

Then, the dissertation has proposed a new methodology for solving robotic task sequencing named A-EH^2C which enables to check the feasibility of reachability,

collision and pose accuracy. A-EH^{2}C has been tested for solving task sequencing of an inspection robots with optical camera system.

The proposed method can be exploited to any industrial robotic system carrying out repetitive tasks; besides, it can be used for sequencing of those tasks which require automatic tool movements, such as navigation problems.

The research opens up new avenues for *dynamic robot programming* which enables real-time robot adjustments.

6.2 Key Findings

Results has showed EH^{2}C is the fastest method with medium number of tasks and appears to be the best method in terms of complexity of computational process as it is only slightly affected by the number of tasks.

Benchmarking studies have highlighted that the complexity of the TSPN solution mostly depends on the following properties of the regions: (1) number; (2) topology; (3) size. For instance, BONMIN suffers when solving 3D geometry because of the topological complexity to compute the optimal tour. Instead, the proposed EH^{2}C method appears to be insensitive to topology. Further investigation is needed to better understand the effect of size.

6.3 Further Works

Further research is clearly needed in the task sequencing domain. This section provides several possible routes for researchers involved in robotic task sequencing.

6.3.1 EH^{2}C

The proposed method for TSPN points out good results, though it presents error fluctuations with several instances. Fluctuation seems to appear when some ellipses are in contact or in overlapping. This should be better investigated and an overlapping logic should be implemented (as in [68]). Further, improving of the hierarchical clustering algorithm could improve solution quality.

Besides, the complexity of the TSPN solution mostly depends on the following properties of the regions: (1) number; (2) topology; (3) size. EH^{2}C method appears to be insensitive to topology. Further investigation is needed to better understand the effect of size.

6.3.2 A-EH^{2}C Attributes

Currently, the proposed method evaluates collisions between end-effector and workpiece. There are other collisions to take into account: end-effector and obstacles; robot and obstacles; end-effector and robot; robot and obstacles; robot and workpiece. An improvement of the collision attribute is required to increase the feasibility of the solution.

This method has a general formulation and can be applied to all sequencing problems. It can be exploited for any industrial robot that carries out tasks or in all navigation systems.

6.3.3 Obstacles

The proposed method has neglected obstacles within robot workspace. Obstacles can be classified in two way: fixed and mobile. The first one could be added as further attribute considering their volume as unfeasible robot space. Instead, obstacle movements require robot real adjustment to avoid collision

The proposed method is overlooking the dynamic reprogramming of the robot which opens up the possibility to real adjust robot according to obstacles movements; additional research is needed to implement this aspect.

6.3.4 Task Sequencing and Path Planning Integration

Researchers are spending time on task sequencing and path planning integration. Kovac [50] introduces a novel model problem: Traveling Salesman Problem with Neighbourhoods and Durative visits (TSP-ND) for task integration. Additional research is needed in this direction as it is still not clear how made a whole integration to generate the optimal path in reasonable time.

6.3.5 Robot Dynamics

The proposed method neglects robot movements as well as motion law; this allow us to have only an approximation of cycle time. Future works should implement further aspects related to robot movements in order to include the time dimension into solution process. Researches are needed in this direction to understand how formalised this aspect.

6.3.6 Task Sequencing Benchmarks

Although there exist instances for TSP, TSPN and CETSP, there is no benchmarks for task sequencing problem. This impedes to compare existing approaches. For this reason, researchers use different case study to test their approaches solving specific case as well as features and constraints. Developing a benchmark for robotic task sequencing problem is an open research question.

References

[1] H. Kagermann, W. D. Lukas, W. Wahlster, “Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution”. VDI Nachrichten, (13) 2011. [Online]. Available: http://www.vdi-nachrichten.com/Technik-esellschaft/Industrie-40-Mit-Internet-Dinge-Weg-4-industriellen-Revolution.

[2] W. MacDougall, Industrie 4.0: smart manufacturing for the future. Germany Trade & Investment (GTAI), 2014.

[3] R. Drath, A. Horch, “Industrie 4.0: Hit or hype?”. [industry forum]. IEEE industrial electronics magazine 8 (2): 56-58, 2014.

[4] “Bosch driving Industry 4.0 forward with innovative products”. 2015. [Online] Available: http://www.bosch-presse.de/pressportal/de/en/bosch-driving-industry-4-0-forward-with-innovative-products-42925.html

[5] N. Wu, X. Li, “RFID Applications in Cyber-Physical System”. [Online]. Available: http://www.intechopen.com/books/deploying-rfid-challenges-solutions-and-open-issues/rfid-applications-in-cyber-physical-system

[6] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, M. Harnisch, “Industry 4.0: The future of productivity and growth in manufacturing industries”. Boston Consulting Group, 14, 2015.

[7] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, M. Hoffmann, M., “Industry 4.0”. Business & Information Systems Engineering 6 (4): 239, 2014.

[8] World Robotics 2016 - Industrial Robots, International Federation of Robotics

[9] J. Shi and S. Zhou, “Quality control and improvement for multistage systems: A survey,” IIE Trans., vol. 41, no. 9, pp. 744–753, Jun. 2009.

[10] P. Franciosa, D. Ceglarek, “Hierarchical synthesis of multi-level design parameters in assembly system”. CIRP Annals – Manufacturing Technology 64: 149-152, 2015.

[11] C. Wurll, D. Henrich, “Point-to-point and multi-goal path planning for industrial robots”. Journal of Robotic Systems 18 (8): 445-461, 2001.

[12] Z. Pan, J. Polden, N. Larkin, S. Van Duin, J. Norrish, “Recent progress on programming methods for industrial robots”. Robotics and Computer-Integrated Manufacturing 28: 87-94, 2012.

[13] K. Trnka, P. Božek, “Optimal motion planning of spot welding robot applications”. Applied Mechanics and Materials 248: 589-593, 2013.

[14] S. Alatartsev, “Robot Trajectory Optimization for Relaxed Effective Tasks”. Ph.D. dissertation, Otto-von-Guericke-Universitat Magdeburg, 2015.

[15] E. Kolakowska, S.F. Smith, M. Kristiansen, “Constraint optimization model of a scheduling problem for a robotic arm in automatic systems”. Robotics and Autonomous Systems 62: 267-280, 2014.

[16] G. J. Woeginger, "Exact algorithms for NP-hard problems: A survey". Combinatorial Optimization-Eureka, You Shrink! Springer Berlin Heidelberg, 2003. 185-207.

[17] Robot Institute of America (RIA), 1979.

[18] ISO 8373, Robots and robotic devices – Vocabulary.

[19] B. Siciliano, L. Sciavicco, L. Villano, G. Oriolo, “Robotics: modelling, planning and control”. Springer Science % Business Media, 2010. ISBN 978-1-84628-641-4.

[20] Z. Ermin, P. Xiaojian, Z. Yabo, “The research of the trajectory of industrial robots modeling simulation based on DELMIA”. Modern Manufacturing Engineering 3, 007, 2013.

[21] Z. Yao, K. Gupta, “Path planning with general end-effector constraints”. Robotics and Autonomous Systems 55: 316-327, 2007.

[22] W. Dong, H. Li, X. Teng, “Off-line programming of Spot-weld Robot for Car-body in White Based on Robcad”. International Conference on Mechatronics and Automation (ICMA): 763-768, 2007.

[23] W. Zhu, W. Qu, L. Cao, D. Yang, Y. Ke, “An off-line programming system for robotic drilling in aerospace manufacturing”. The International Journal of Advanced Manufacturing Technology 68 (9-12): 2535-2545, 2013.

[24] E. Galceran, M. Carreras, “A survey on coverage path planning for robotics”. Robotics and Autonomous Systems 61: 1258-1276, 2013.

[25] A. F. Nicolescu, A. M. Ivan, “Advanced off-line programming and simulation of a flexible robotic manufacturing cell using Kawasaki Pc-Roset”. Proceedings in Manufacturing Systems 6 (4): 239-244, 2011.

[26] Z. Wang et al., “Off-line programming of Spot-Weld Robot for Car-Body in White Based on Robcad”. International Conference on Mechatronics and Automation, China, 2007.

[27] M. Jünger, G. Reinelt, G. Rinaldi. "The traveling salesman problem". Handbooks in operations research and management science 7: 225-330, 1995.

[28] D.S. Johnson, L.A. McGeoch, “The traveling salesman problem: A case study in local optimization”. Local search in combinatorial optimization 1: 215-310, 1997.

[29] L. Gueta, R. Chiba, J. Ota, T. Ueyama, T. Arai, “Coordinated motion control of robot arm and a positioning table with arrangement of multiple goals”. IEEE International Conference on Robotics and Automation: 2252-2258, 2008.

[30] G. Laporte, Y. Nobert, “Generalized travelling salesman problem through n sets of nodes: An integer programming approach”. INFOR: Information Systems and Operational Research 21 (1): 61-75, 1983.

[31] M. Saha, T. Roughgarden, J. Latombe, G. Sanchez-Ante, “Planning tours of robotic arms among portioned goals”. The International Journal of Robotics Research 25 (3): 207-223, 2006.

[32] C. Wurll, D. Henrich, H. Worn, “Multi-Goal Path Planning for Industrial Robots”. In International Conference on Robotics and Application, 1999.

[33] E. M. Arkin, R. Hassin, “Approximation algorithms for the geometric covering salesman problem”. Discrete Applied Mathematics 55 (3): 197-218, 1994.

[34] S. Alatartsev, V. Mersheeva, M. Augustine, F. Ortmeier, “On optimizing a sequence of robotic tasks”. In IEEE/RSJ International conference on intelligent robots and systems (IROS), 2013.

[35] E. M. Arkin, R. Hassin, “Approximation algorithms for the geometric covering salesman problem”. Discrete applied mathematics 55 (3): 197-218, 1994.

[36] K. Elbassoni, A. V. Fishkin, R. Sitters, “Approximation algorithms for the Euclidean traveling salesman problem with discrete and continuous neighborhoods”. International journal of computational geometry and applications 19 (2): 173-193, 2009.

[37] I. Gentilini, K. Nagamatsu, K. Shimada, “Cycle time based multi-goal path optimization for redundant robotic systems”. In IEEE International Conference on Intelligent Robots and Systems (IROS), 2013.

[38] K. Vicencio, B. Davis, I. Gentilini, “Multi-goal path planning based on the generalized traveling salesman probleneighborhoods”. In IEEE/RSJ International conference on intelligent robots and systems (IROS), 2014.

[39] K. Vicencio, T. Korras, K. A. Bordignon, I. Gentilini, “Energy-optimal path planning for six-rotors on multi-target missions”. In IEEE/RSJ International conference on intelligent robots and systems (IROS): 2481-2487, 2015.

[40] S. Alatartsev, S. Stellmacher, F. Ortmeier, “Robotic task sequencing problem: a survey”. Journal of Intelligent & Robotic Systems 80 (2): 279-298, 2015.

[41] A. Kovacs, “Integrated task sequencing and path planning for robotic remote laser welding”. International Journal of Production Research 54 (4): 1210-1224, 2016.

[42] J. Gudmundsson, C. Levcopoulos. "Hardness result for TSP with neighborhoods", 2000.

[43] W. Mennell, “Heuristic for solving three routing problems: close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem”. Ph.D. dissertation, University of Maryland, 2009.

[44] M. Dror, A. Efrat, A. Lubiw, J. S. Mitchell, “Touring a sequence of polygons”. In Proceedings of the thirty-fifth annual ACM symposium on Theory of computing: 473-482, 2003.

[45] S. Alatartsev, M. Augustine, F. Ortmeier, “Constricting Insertion Heuristic for Traveling Salesman Problem with Neighborhoods”. In 23rd International Conference on Automated Planning and Scheduling (ICAPS), 2013.

[46] I. Gentilini, F. Margot, K. Shimida, “The travelling salesman problem with neighbourhoods: MINLP solution”. Optimization Methods and Software 0: 1-15, 2011.

[47] H. Chen, T. Fuhlbrigge, X. Li, “Automated industrial robot path planning for spray painting process: a review”. In Automation Science and Engineering: 522-527, 2008.

[48] C. Hofner, G. Schmidt, “Path planning and guidance techniques for an autonomous mobile cleaning robot”. Robotics and autonomous systems 14: 199-212, 1995.

[49] X. Wang, Y. Shi, D. Ding, X. Gu, “Double global optimum genetic algorithm–particle swarm optimization-based welding robot path planning”. Engineering Optimization 48: 299-316, 2016.

[50] A. Kovac, “A task sequencing for remote laser welding in the automotive industry”. International Conference on Automated Planning and Sceduling (ICAPS): 457-461, 2013.

[51] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, D. Lane, “Path planning for autonomous underwater vehicles”. IEEE Transactions on Robotics 23: 331-341, 2007.

[52] S.N. Spitz, A. A. Requicha, “Multiple-goals path planning for coordinate measuring machines”. IEEE International Conference on Robotics and Automation (ICRA) 3, 2322-2327, 2000.

[53] P. Zacharia, E. Xidias, N. Aspragathos, “Task scheduling and motion planning for an industrial manipulator”. Robotics and Computer-Integrated Manufacturing 29: 449-462, 2013.

[54] Y. Huang, L. B. Gueta, R. Chiba, T. Arai, T. Ueyama, J. Ota, “Selection of manipulator system for multiple-goal task by evaluating task completion time and cost with computational time constraints”. Advanced Robotics 27 (4): 233-245, 2013.

[55] Ang, Marcelo H., Lin Wei, and Lim Ser Yong. "An industrial application of control of dynamic behavior of robots-a walk-through programmed welding robot". ICRA'00. Robotics and Automation, IEEE International Conference on. Vol. 3: 2352-2357, 2000.

[56] G.Oriolo, M. Vendittelli, “A control-based approach to task-constrained motion planning”. IEEE International Conference on Intelligent Robots and Systems: 297-302, 2008.

[57] G. Reinhart, U. Munzert, W. Vogl, “A programming system for robot-based remote-laser-welding with conventional optics”. CIRP Annals-Manufacturing Technology 57: 37-40, 2008.

[58] Y. K. Hwang, N. Ahuja, “Gross motion planning: a survey”. ACM Computing Surveys (CSUR) 24 (3): 219-291, 1992.

[59] A. K. Jain, “Data clustering: 50 years beyond K-means”. Pattern recognition letters 31 (8): 651-666.

[60] M. G. Steinbach, V. Kumar, “A comparison of document clustering techniques”. In KDD workshop on text mining 400 (1): 525-526, 2000.

[61] G. Reinelt, "TSPLIB—A traveling salesman problem library". ORSA journal on computing 3 (4): 376-384, 1991.

[62] X. Pan, F. Li, R. Klette, “Approximate shortest path algorithms for sequences of pairwise disjoint simple polygons” Canadian Conference on Computational Geometry: 175-178, 2010.

[63] F. Li, K. Reinhard, "Rubberband algorithms for solving various 2D or 3D shortest path problems". In Computing: Theory and Applications, ICCTA'07. International Conference on, 9-19. IEEE, 2007.

[64] S. K. Saren, V. Tiberiu, “Review of flexible manufacturing system based on modeling and simulation”. Annals of the university of Oradea, Fascicle of management and technological engineering issue #1, may 2016.

[65] I. Gentilini, F. Margot, K. Shimada, “STSPN Instances”. URL http://wpweb2.tepper.cmu.edu/fmargot/ampl.html

[66] Alatartsev, S.; Augustine, M.; and Ortmeier, F. 2012. https://cse.cs.ovgu.de/cse/traveling-salesman-problem-with-neighborhoods-tspn/

[67] D. Ceglarek, P. Franciosa, “RLW Door Simulation 33 slides”. [Online]. https://www.researchgate.net/publication/275408808_Ceglarek_Franciosa_RLW_Door_Simulation_33_slides.

[68] J. Faigl, V. Vonasek, L. Preucil, “A multi-goal path planning for goal regions in the polygonal domain”. In European Conference on Mobile Robots (ECMR): 171-176, 2011.