Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 72 (2018) 386–391

www.elsevier.com/locate/procedia

51st CIRP Conference on Manufacturing Systems

Optimal task positioning in multi-robot cells, using nested meta-heuristic swarm algorithms

G. Nicola[a,*], N. Pedrocchi[a], S. Mutti[a], P. Magnoni[a], M. Beschi[a]

[a]*Consiglio Nazionale delle Ricerche, Institute of Industrial Technologies and Automation, via A. Corti 12, 20133 Milan, Italy*

[*] Corresponding author. *E-mail address:* giorgio.nicola@itia.cnr.it

Abstract

Process planning of multi-robot cells is usually a manual and time consuming activity, based on trials-and-errors. A co-manipulation problem is analysed, where one robot handles the work-piece and one robot performs a task on it and a method to find the optimal pose of the work-piece is proposed. The method, based on a combination of Whale Optimization Algorithm and Ant Colony Optimization algorithm, minimize a performance index while taking into account technological and kinematics constraints. The index evaluates process accuracy considering transmission elasticity, backslashes and distance from joint limits. Numerical simulations demonstrate the method robustness and convergence.

Peer-review under responsibility of the scientific committee of the 51st CIRP Conference on Manufacturing Systems.

Keywords: Task placement; accuracy indexes; optimization algorithms; industrial robotics;

1. Introduction

A robotic dual-arm setup is a workcell layout more and more used in industrial tasks. Specifically, in many tasks, one robot, hereafter Robot 1, performs the task on the workpiece (e.g. welding, machining, painting, *etc.*) while the second robot, hereafter Robot 2, is holding the workpiece in a static position. In such a scenario, the identification of the Robot-2 configuration in term of joint positions is fundamental to achieve a good final result. Indeed, the robot performance largely changes over the workspace, making important where to execute the task within the workspace.

For a single-arm arm setup, this problem is generally tackled by the maximization of dexterity along the path [3–6]. However, such indexes do not take into account mechanical task-dependent problems such as backlashes or transmissions elasticity that cause poor robot accuracy even after static calibration. To overcome such limitations, many works [1,2,8,9] propose objective functions tailored on the specific robotic task and robotic setup. The adopted optimization methods are many and different: non-linear programming techniques [1], genetic algorithms [7], cycle time minimization methods [8,9], as well as joint torques/energy minimization [2,7].

In comparison to one-arm application, cooperative robots introduce some issues: (i) high number of degrees of freedom (DOFs) with many internal constraints; (ii) once the Cartesian reference frame of the task is determined (i.e., the pose of Robot-2 when holding the part), the transformation between task frame and Robot 1 joint positions is not unique, e.g. for every task point a set of joint angles is possible (i.e. robot configurations) since the inverse kinematics is not a bijective function. Even infinite solutions may be possible if the task is lazy-constrained (e.g., the tool can rotate around its own axis). In literature, three different approaches have been presented:

i the two robots are considered as a unique set of variables, and non-linear optimization methodologies are run over the complete space of solutions. In [8], the optimization is performed by a gradient method, that, however, may lead to the identification of local minima, without the possibility to span all the alternative robot configurations.

ii velocity constraints are relaxed [1], i.e., the robot is supposed to be able to change configuration between two consecutive nodes. Pamanes et al. [1] proposes a non-linear programming technique requiring at the beginning a candidate solution that satisfies all the constraints. The following limitations exist: (i) the approach is not full-automatic and (ii) it is not suitable for dual arm applications where the problem is computationally complex.

iii one of the Robot-1 configurations is forced [2,7,9], making bijective the inverse kinematics function. Specifically, Santos et al. [2] proposes a tunneling method for searching the global minimum, but since the algorithm numerically solves a highly non linear equation many times, it is not suitable for all those application in which the computational time is as relevant. In [7], instead, the optimization is performed by two consecutive genetic algorithms, making complex the balancing between exploration and exploitation. Karamani et al. [9] proposes a response surface

2212-8271 © 2018 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the 51st CIRP Conference on Manufacturing Systems.
10.1016/j.procir.2018.03.081

method but, since the objective function and the output are interpolated, the optimal solution might be very far from the global minimum because of the inaccuracy introduced by interpolation model adopted.

In order to overcome the aforementioned limitations, this work proposes to split the problem in two sub-problems, and to run iteratively two nested optimizers: first the problem of the object positioning is solved (e.g., definition of the Robot-2 holding position), then, the Robot-1 configurations are optimized. Finally, an iteration over the two steps is computed. Such method allows a decoupling of the problem in two easier sub-problems, and the adoption of different optimization methodologies for each of the two steps. Among the plenty of optimization algorithms, a Whale Optimization Algorithm (WOA) and an Ant Colony Optimization algorithm (ACO) have been selected for the first and second optimization steps respectively. Specifically, the use meta-heuristic algorithms was decided since these algorithms typically perform well with a large set of mathematical problems and have a good balance between solution accuracy and calculation time. On the one hand, Mirjalili et al. [10] prove that WOA is among the best-performing meta-heuristic algorithms on a large set of mathematical problems. On the other hand, the ACO is extremely efficient when a combinatory problem has to be solved [12].

The paper is organized as follow: in Section 2, an iterative two-step optimization methodology to optimize the motion-coordination of two robots is described; in Section 3, the analysis of the method performance is shown; finally, in Section 4 conclusions and future developments are pointed out.

2. Whale and Ant Colony Optimization

2.1. Overall strategy

The problem is solved by two nested optimizers: first, a Whale Optimization Algorithm (WOA) optimizes the starting node pose, then, an Ant Colony Optimization algorithm (ACO) finds the best robot configurations at each WOA step.

The input data required by the WOA are the robotic cell geometry, the kinematic model of the robots, and the task description (path and execution time), while the output is the Robot 1 joint position corresponding to the task starting position. Specifically, the WOA is a meta-heuristic swarm optimization where each agent, called "whale", evaluates the objective function from the input data and the candidate solution, i.e., task starting position.

Given the Robot-1 joint position of the starting node, the task can be modeled as a set of Cartesian reference frames, hereafter nodes, that the Robot-1 should be follow during the task execution. Furthermore, once the starting point of the task is computed, also the holding point pose for Robot 2 is consequently imposed. Once all the nodes and the Robot-2 pose are determined, it is possible to evaluate the constraints:

1. to check the reachability of each node by Robot 1 with at least one valid configuration, i.e., at least one joint solution of the inverse kinematic problem exists;
2. to check the reachability of the task holding point by Robot 2 with at least one valid configuration, i.e., at least one joint solution of the inverse kinematic problem exists;
3. to check the existence of at least one continuous path in the joint space that grants the dynamics (velocity and accelerations) constraints.

Algorithm 1 WACO

```
Load Task and robotic cell kinematics
function WOA(Kinematics)
    while iter ≤WOA max iterations do
        for (each whale) do
            Calculate the task starting frame as in (6)
            Calculate all task frames and holding pose by (7) (8)
            Verify reachability of the task by Robot 2 (10)
            Verify reachability of the task by Robot 1 (9)
            function ACO(Joint points)
                Calculate optimal Robot 1 configurations
            end function return Optimal Robot 1 configurations
            Interpolate joint angles with continuous spline
            Verify instantaneous joint speed limits (13)
            Calculate objective function the task with (4)
        end for
        iter = iter + 1
    end while
end function return Optimal task position and robot joints
```

Since each node may have multiple (even infinite) equivalent joint configurations, the last step consists in finding the shortest path over an oriented graph using ACO.

Once the optimal path is found, a scalar cost function is calculated over the path. The implemented cost function is a tuple of four indexes: joint speed, joint acceleration, distance from joint limits and number of time the speed joint reaches zero, and it will be described in the next paragraph.

The overall methodology has been called Whale and Ant Colony Optimization (WACO) and its own pseudo code is in the table Algorithm 1. The WACO is intended to be computationally efficient to reach an acceptable trade-off between the required computation time and the solution accuracy. Therefore, it is worth to note that the number of the nodes along the path has to be kept limited. Indeed, a large number of the nodes may reduce dramatically the performance of the ACO algorithm. However, this limitation is not critical since the problem of the minimization of the length of the joint-trajectory is locally linearizable. Therefore, a low discretization of the path (a node each some centimeters) does not introduce a large error in the optimal path identification. Finally, to have a more precise evaluation of the cost function, the optimal path computed by ACO can be oversampled through a simple fitting of the approximate trajectory using the splines.

2.2. Problem Formalization

Referring to Fig 1, denote the following variables:

$\mathbf{T}_b^a$	Transformation from frame $\{b\}$ to $\{a\}$.
$\mathbf{q}_{R_i} \in \mathbb{R}^{dof_i}$	vector of joint angles of Robot i-th
$\{R_i\}$, $\{tool_i\}$	Base and Tool Frame of Robot i-th
$\{h\}$	Object Holding Frame, i.e., the pose of the end-effector of Robot-2, $\{h\} \equiv \{tool_2\}$
$\{t_k\}$	Work Object Frame of the k-th node of the trajectory, i.e., the pose the end-effector of Robot-1 should match, $\{t_k\} \equiv \{tool_1\}$.
$\left\{\mathbf{T}_{R1}^{t_k}\right\}_{k=0,\dots,N_p}$	N_p ordered nodes that Robot 1 have go trough

Furthermore, denote $\mathbf{T}_{R_i}^{tool_i} = FK_{R_i}(\mathbf{q}_{R_i})$ and $Q_{R_i} = IK_{R_i}(T_{tool_i}^{R_i})$ as the forward and inverse kinematics respectively. Specifically, $Q_{R_i} = \{\mathbf{q}_{R_1,s} : s = 1,\dots,N_{sol}\}$ is the set of all the possible $N_{sol} \in \mathbb{N}^+$ solutions of the inverse kinematics. There-

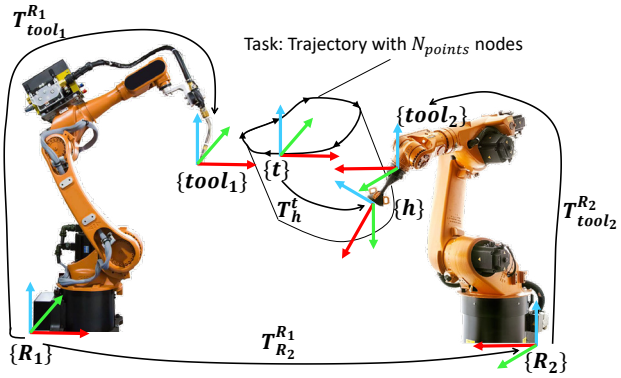


Fig. 1: Robotic cell scheme

fore the map $\{Q_{R_1,k}\}_{k=0,\dots,N_p}$ stores all the feasible Robot-1 configurations along the N_p points of the path. The dimension of such map may be large, accordingly to the number of the points N_p and the feasible IK solutions in each point N_{sol_k}.
Finally, within the map $\{Q_{R_1,k}\}_{k=0,\dots,N_p}$ it is possible to identify a large number N_{feas} of different feasible path $\mathcal{P}_l = \{\mathbf{q}_{R_1,s,k} : s \in 1,\dots,N_{sol_k}\}_{k=0,\dots,N_p}$ with $l = 1,\dots,N_{feas}$, such that the Robot-1 can move from the first point of the trajectory to the last one without any change of the configuration.

2.3. Objective function

In this work only three parameters have been considered as proxies of the robot execution accuracy: (i) Transmission elasticity; (ii) Backlashes; (iii) Distance from joint limits. Extension to more parameters is however simple, and it can be customized on the actual performance and characteristics of the robotic setup.

Transmission elasticity. To reduce its effect, it is necessary to minimize the inertial torques on the joints. However, the use of a physical model of the robot can be computationally too expensive. As consequence, joints acceleration and speed (for non-linear torques) are used as torques estimators. Therefore, given a feasible path $\mathcal{P}$, quality indexes are introduced.

$$i_{speed} = \sum_{k=0}^{N_p}\sum_{j=1}^{dof_1}\left(\dot{q}^j_{R_1,k}\right)^2 \qquad i_{acc} = \sum_{k=0}^{N_p}\sum_{j=1}^{dof_1}\left(\ddot{q}^j_{R_1,k}\right)^2. \tag{1}$$

Backlashes. Backlashes decrease the accuracy whenever a joint speed changes sign. Therefore, the number of times the joint speed reaches zero can be used as a proxy to estimate the effects of the backlashes. Denoting ν as a function counting the velocity inversions along the robot path $\mathcal{P}$, the backlashes-quality index results:

$$i_{inv} = \nu(\mathcal{P}). \tag{2}$$

Distance from joint limits. In general, industrial manipulators have lower performances in proximity of joint limits. Therefore, given a feasible path $\mathcal{P}$, we introduce the quality index

$$i_j = \prod_{k=0}^{N_p}\prod_{j=1}^{dof_1}\left(1 + \left(\frac{q^j_{R_1,k} - \bar{q}^j_{R_1}}{(q^j_{R_1,k} - q^{j+}_{R_1})(q^j_{R_1,k} - q^{j-}_{R_1})}\right)^2\right). \tag{3}$$

$q^{j+}_{R_1}$ and $q^{j-}_{R_1}$ are the upper and lower joint limits of the j-th joint of Robot 1 while $\bar{q}^j_{R_1}$ is the mean joint range value.

Finally all these indexes are combined in the following scalar multi-objective function f_{obj}:

$$f_{obj} = i_{speed} i_{acc} i_{inv} i_j \tag{4}$$

2.4. The optimization problem

The objective function f_{obj} depends only on Robot 1 joints values along a feasible path $\mathcal{P}_l$, while the starting node pose $\mathbf{T}^{t_0}_{R1}$ is defined accordingly to the Robot-2 tool pose $\mathbf{T}^{R_2}_h$.
Given the physics of the problem, h is rigidly connected to t_0 if existing. Therefore, the optimization problem can be reduced to a two-step optimization of the Robot-1 configuration: first, a feasible $\mathbf{q}_{R1_0}$ granting the reachability limits of the Robot-2 is computed, then, it is possible to calculate the path $\mathcal{P}_l|_{\mathbf{T}^h_{R_2}}$ that minimizes the objective function.
Summarizing, the mathematical formulation of the problem is:

$$\begin{aligned}
&\text{minimize} && f_{obj} \\
&\text{subject to} && \begin{cases}
\mathbf{q}_{R1_0} : \exists\, \mathbf{q}_{R_1,k} = IK_{R_1}\left(\mathbf{T}^{t_k}_{R_1}\right),\ \forall k \in 1,\dots,N_{point} \\
\mathbf{q}_{R1_0} : \exists\, \mathbf{q}_{R_2} = IK_{R_2}\left(\mathbf{T}^{h}_{R_2}\right) \\
q^{j-}_{R_1} \le q^j_{R_1,k} \le q^{j+}_{R_1} \quad \forall\, j \in 1,\dots,dof_1 \\
\dot{q}^{j-}_{R_1} \le \dot{q}^j_{R_1,k} \le \dot{q}^{j+}_{R_1} \quad \forall\, j \in 1,\dots,dof_1 \\
q^{j-}_{R_2} \le q^j_{R_2} \le q^{j+}_{R_2} \quad \forall\, j \in 1,\dots,dof_2
\end{cases}
\end{aligned} \tag{5}$$

2.5. The method

For each WOA member, the starting Robot 1 joint positions $\mathbf{q}_{R_1,0}$ is evaluated and the task starting frame is calculated as:

$$\mathbf{T}^{t_0}_{R_1} = FK_{R_1}(\mathbf{q}_{R_1,0}) \tag{6}$$

Thus, each node $\mathbf{T}^{t_k}_{R_1}$ of the path and $\mathbf{T}^h_{R_2}$ can be referred to the starting pose as:

$$\mathbf{T}^{t_k}_{R_1} = \mathbf{T}^{t_0}_{R_1}\mathbf{T}^{t_k}_{t_0} \quad k \in 1,2,\dots,N_p \tag{7}$$

$$\mathbf{T}^h_{R_2} = \mathbf{T}^{R_1}_{R_2}\mathbf{T}^{t_0}_{R_1}\mathbf{T}^h_{t_0} \tag{8}$$

with $\mathbf{T}^{t_k}_{t_0}$ and $\mathbf{T}^h_{t_0}$ constant transformation given by the geometry of the tools. After the calculation of all the nodes, the analytic inverse kinematics is computed for each node and all the solutions are stored,

$$Q_{R_1,k} = IK_{R_1}\left(\mathbf{T}^{t_k}_{R_1}\right) \quad k \in 1,2,\dots,N_p \tag{9}$$

$$Q_{R_2,h} = IK_{R_2}\left(\mathbf{T}^h_{R_2}\right) \tag{10}$$

So, if all the nodes are reachable with at least one valid configuration and the holding point is reachable by Robot 2, it is possible to continue to the next step (the optimization of Robot 1 configurations). Otherwise, the starting joint position $\mathbf{q}_{R_1,0}$ is discarded.
The problem of finding the optimal combination of robot configurations is essentially a "shortest path problem" on a oriented graph [11]. Indeed, all the solutions of the IK about nodes

can be arranged on a graph. Each graph layer corresponds to a task point and each graph node is a valid configuration for a task point. Then, all nodes belonging to the same layer are linked unidirectionally with all nodes of the successive layer. So, ACO (Ant Colony Optimization) was used as optimization algorithm. This formulation allows the robot to change configuration along the path, even if it is not convenient because it would require joint trajectories with high levels of speed saturation. Remarkably, when the Robot-1 has to follow complex Cartesian path, it might be necessary to change Robot-1 configuration to successfully complete the task. However, the chosen heuristic discourages these changes of configurations unless not strictly necessary. The chosen heuristic is the inverse of joint squared distance between each point, as shown in (11).

$$heuristic(k) = \frac{1}{\sum_{i=1}^{dof_1}\left(q^j_{R_1,k} - q^j_{R_1,k-1}\right)^2} \quad k \in 1,\dots,N_p \tag{11}$$

The cost function for the pheromone deposition is the total squared joints distance along the path. Moreover, at each segment is verified that the joints mean speed is not above the joints speed limits. The cost function at every point is shown in (12).

$$\begin{cases} cost(k) = \sum_{i=1}^{dof_1}\left(q^j_{R_1,k} - q^j_{R_1,k-1}\right)^2 & \dot{\mathbf{q}}^-_{R_1} \le \dot{\bar{\mathbf{q}}} \le \dot{\mathbf{q}}^+_{R_1} \\ cost(k) = inf & \dot{\bar{\mathbf{q}}} > \dot{\mathbf{q}}^+_{R_1} \vee \dot{\bar{\mathbf{q}}} < \dot{\mathbf{q}}^-_{R_1} \end{cases} \tag{12}$$

Then the output of the ACO is Q the optimal set of Robot 1 joint positions for all the k-th nodes $k \in 1,\dots,N_p$.
In order to use instantaneous values of speed and accelerations in the evaluation of the objective function, the joints positions at each nodes are interpolated with a set of continuous smoothing splines (i.e. 3^{rd} grade polynomial curves). Finally, it is possible to verify that even the joints speed are within the acceptable range (13) and it is possible to calculate the objective function, as in Section 2.3.

$$\dot{q}^{j-}_{R_1} \le \dot{q}^j \le \dot{q}^{j+}_{R_1} \quad j = 1,2,\dots,dof_1 \tag{13}$$

3. Performance Analysis

3.1. Experimental setup

The setup is composed by two ABB IRB4600 20.5 (Fig.1) and it is configured for cooperative welding application.
The results of two benchmark tasks are here reported: Task 1 (Fig. 2) consists on performing a circular trajectory of radius 200 mm with a constant linear speed of 100 mm/s discretized in 20 nodes; Task 2 (Fig. 3) consists on a 3D L-shaped trajectory with a linear speed of 100 mm/s discretized in 20 nodes.
Furthermore, different optimization slots have been launched with different parameters. Specifically,

- WOA iterations [50, 100, 200, 300, 400, 500, 600]
- Whales number [20, 30, 40, 50, 60]
- WOA "optimization/exploration" $A_{lim} = 1, 0.5, 0.2, 0.1$

The parameters of ACO are instead kept constant for every combination because it has been noticed that they are not critical for the convergence. The chosen parameters are:

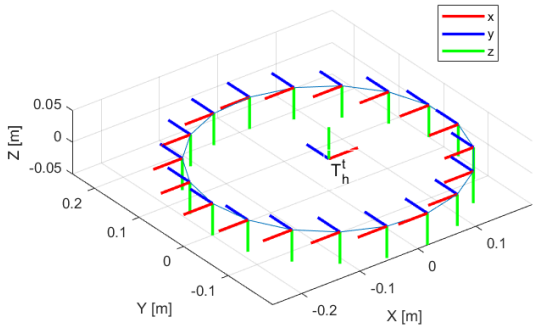


Fig. 2: Task 1

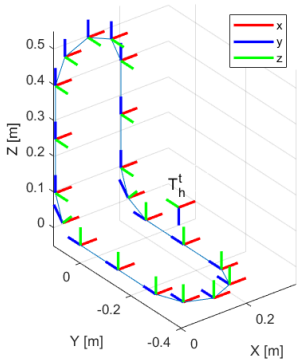


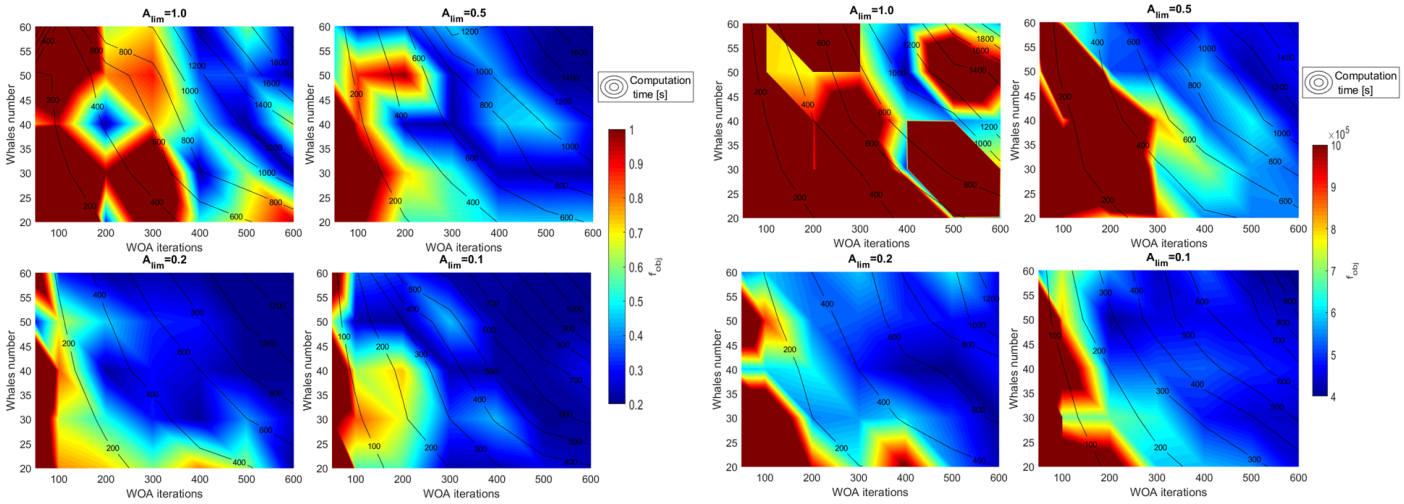
Fig. 3: Task 2

- ACO iterations 30
- Ant number 10

For each combination of the parameters, a set of 10 optimizations has been performed in order to have a statistic sample. For every set of optimizations, mean value and standard deviation of the optimal solution is calculated. Moreover, the mean value of the computational time and the mean value of the objective function are calculated. The analysis has been performed on Matlab R2017a with a desktop computer with a CPU Intel i7-7700 (3.8 GHz) and 8 GB RAM.

3.2. Results analysis

In Fig. 4a, the mean optimum value of the optimization index for the task position is shown together with the mean value of the computational time. Comparing the results, it can be easily noticed that better performances are achieved by decreasing the value of A_{lim}, i.e. increasing the amount of iterations spent exploring the research domain. Indeed, the research domain is limited only by the joint limits, although, the "feasible" domain, where the trajectory is feasible, is much smaller. As a consequence, many trials are necessary to find a feasible solution and in particular a good solution to be optimized. Moreover, the objective function, as designed in Section 2.3, has a low gradient far away from the joint limits. So, the variation of the objective function is low and the optimization is hard to perform.
Then, it is possible to analyse the required computation time, shown in Fig. 4a with black lines representing "iso-computation time" simulations. It can be noticed that the computational time decrease as A_{lim} decreases. Indeed, the greatest part of the computation time is spent by ACO and by the evaluation of the objective function, including the interpolation of joints trajectories. Although these two step are performed only when there is a high possibility that the evaluated trajectory is feasible. The simulations with low optimization/exploration ratio (i.e. A_{lim} low) have the highest ratio of infeasible trajectories over total



(a) Task 1 (b) Task 2

Fig. 4: Average optimum values for Task 1 and Task 2

Table 1: Dispersion of optimal task starting positions with 600 iterations and 60 whales

Robot 1 Joint			1	2	3	4	5	6
Task 1	Joint pos.[rad]	mean	-0.059	0.697	-0.514	-0.065	-0.848	-0.566
		std	0.170	0.079	0.056	0.081	0.053	0.705
Task 2	Joint pos.[rad]	mean	-0.107	0.110	-0.096	-1.145	-0.692	-0.527
		std	0.340	0.111	0.125	0.850	0.870	0.410

number of trajectories evaluated. Therefore, comparing simulations with the same number of evaluated trajectories, the simulation with low A_{lim} reach less frequently the phases of the ACO and of the calculation of the objective function less times. The we can affirm that the "iso-computation time" curves represent nothing else than curves with the same number of feasible trajectories evaluated.

Studying instead the optimal solution, e.g. the joints position in the starting node, we notice that the convergence is less evident and for some joints the standard deviation is very high. In Table 1, it is shown the dispersion of the optimal solution for the following set of parameters: 600 iterations, 60 whales and $A_{lim} = 0.2$. It can be easily noticed that for Task 1 the standard deviation of the 6^{th} joint is more than 10 times higher than 3^{rd} joints standard deviation, this result can be explained by the low gradient of the objective function. Although, from a physical point of view this results leads to 2 conclusions, first of all there is a region of joint space where the task is performed optimally instead of a single point. Secondly, depending on the task trajectory some joints are more relevant than others for optimality and feasibility.

Analysing the results for Task 2 we can notice many similarities, first of all as for Task 1 decreasing A_{lim} the performances increase both as optimality and as computation time. It can also be noticed that the improvements on both aspect are higher than for Task 1. Thanks to increased complexity of trajectory that "feasible" domain is even smaller than for Task 1, so the ratio of infeasible solutions is higher. Confronting the dispersion of the solution obtained with 600 iterations, 60 whales and $A_{lim} = 0.1$ in Table 1 we notice higher values of standard deviation compared to Task 1. Indeed, having a higher ratio of infeasible solutions, the optimization with the same number of trajectories evaluated has worse performance. Although it can be noticed that, as for Task 1, some joints have much lower standard deviation because they are the main responsible for the solution feasibility and optimality.

4. Conclusions and future developments

In this paper a novel method named WACO (Whale and Ant Colony Optimisation) to optimise task placement for process planning of multi-robot cells is presented, the optimisation is performed by 2 nested meta-heuristic algorithms (WOA and ACO) in order to enhance the overall accuracy of the task. The method assures the kinematic feasibility of the task trajectory although the collision avoidance is not implemented yet. The WACO has proven to converge to an optimal solution but the calculation time is still not completely satisfying. Although, the method is highly parallelizable so the calculation time is expected to decrease.

As future developments the collision avoidance will be implemented, the method's code will be updated in order to allow the parallel computation and finally an experimental campaign will be conducted to verify the objective function goodness.

References

[1] Pamanes, G., Zeghloul, S.. Optimal placement of robotic manipulators using multiple kinematic criteria. Proceedings - IEEE International Conference on Robotics and Automation 1991;1(April):933–938.

[2] dos Santos, R.R., Steffen, V., Saramago, S.d.F.. Optimal Task Placement of a Serial Robot Manipulator for Manipulability and Mechanical Power Optimization. Intelligent Information Management 2010;02(09):512–525.

[3] Yoshikawa, T.. Manipulability and redundancy control of robotic mechanisms. Proceedings 1985 IEEE International Conference on Robotics and Automation 1985;2:1004–1009.

[4] Angeles, J., López-Cajún, C.S.. Kinematic Isotropy and the Conditioning Index of Serial Robotic Manipulators. The International Journal of Robotics Research 1992;11(6):560–571.

[5] Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.. Robotics, Modelling, Planning and Control. Springer; 2009.
[6] Legnani, G., Tosi, D., Fassi, I., Giberti, H., Cinquemani, S.. The point of isotropy and other properties of serial and parallel manipulators. Mechanism and Machine Theory 2010;45(10):1407 – 1423.
[7] Vosniakos, G.C., Matsas, E.. Improving feasibility of robotic milling through robot placement optimisation. Robotics and Computer-Integrated Manufacturing 2010;26(5):517–525.
[8] Feddema, J.T.. Kinematically optimal robot placement for minimum time coordinated motion. IEEE International Conference on Robotics and Automation Minneapolis 1996;(April).
[9] Kamrani, B., Berbyuk, V., Wäppling, D., Stickelmann, U., Feng, X.. Optimal robot placement using response surface method. The International Journal of Advanced Manufacturing Technology 2008;44(1-2):201–210.
[10] Mirjalili, S., Lewis, A.. The Whale Optimization Algorithm. Advances in Engineering Software 2016;95:51–67.
[11] Cormen, T.H.. Introduction to algorithms. MIT press; 2009.
[12] Dorigo, M., Birattari, M., Stutzle, T.. Ant colony optimization. IEEE Computational Intelligence Magazine 2006;1(4):28–39.
[13] Dorigo, M., Maniezzo, V., Colorni, A.. Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 1996;26(1):29–41.

Appendix A. WOA Whale Optimization Algorithm

In this section, the Whale Optimization Algorithm (WOA) is briefly presented. For a deeper analysis refers to [10].
The algorithms, compared to other well established meta-heuristics algorithms, e.g. the PSO (particle Swarm Optimization), is very competitive [10]. It is able to solve a wide range of optimization problems such as uni-modal and multi-modal or with a high number of variables. Moreover, the algorithms allows the user set the ratio between exploration and exploitation. The WOA is a swarm meta-heuristic algorithm that takes inspiration from the hunting strategy of the humpback whales, so 3 heuristics have been developed: the encirclement, the bubble-net attacking method and the exploration.

At each iteration the algorithm chooses between the heuristics by means of two coefficients A and p. A absolute value decrease linearly from 2 to 0, as the WOA iterations reaches the maximum, meanwhile is modified by a random component, instead p is a random number in $[0, 1]$. So the heuristic is chosen as: $p < 0.5$ and $|A| < A_{lim}$ Encirclement; $p \geq 0.5$ Bubble-net attacking method; $p < 0.5$ and $|A| \geq A_{lim}$ Exploration.

The "Encirclement" heuristic mimics the encirclement of the prey so all the whales will position inside an hypercube, the number of dimensions of the hypercube are the number of variables to optimize, whose side is at maximum the distance between the whales and the best so far solution. Moreover as the iterations increases the the hypercube will shrink.
The "Bubble-net attacking method" consist in an helix shrinking shaped movement around the prey so the whales will be positioned on a logarithmic spiral centered on the best so far solution, even the size of the spiral tends to decrease as the algorithm reaches the maximum iterations.
The "Exploration" is performed similarly to the "Encirclement", although the hypercube instead of being centered on the best so far solution, is centered on a random chosen whale.
The pseudo code of the WOA is in Alg. 2. The algorithm uses only two heuristics at the same moment. Essentially it depends on A_{lim} and the iteration reached by the algorithm. In the firsts iterations $|A| \geq A_{lim}$ so the algorithm will be more focused on the exploration of the search domain, meanwhile close to the maximum number of iterations $|A| < A_{lim}$ so the algorithm optimize the best solution found. It is possible to set the amount of iterations spent exploring by setting the value of A_{lim}.

Algorithm 2 WOA

```
Initialize whales positions
Calculate fitness of every whale
while t < max iteration do
    for each whale do
        Update algorithm parameters
        if p < 0.5 then
            if |A| < A_lim then
                Encirclement
            else
                Exploration
            end if
        else
            Bubble-net attacking method
        end if
    end for
    Check if any whale is beyond the search space and correct it
    Calculate whales fitness
    Update fittest whale
    t = t + 1
end while
```

Algorithm 3 ACO

```
Initialize pheromone on the graph
while t < max iterations do
    for each ant do
        for each graph layer do
            Choose next path node with (B.1)
        end for
        Calculate cost on path
    end for
    Depose pheromone on the graph
    t = t + 1
end while
```

Appendix B. ACO Ant Colony Optimization

Ant Colony Optimization (ACO) is a family of well established swarm meta-heuristic algorithms for graph research. In particular, each ant deposes a certain amount of pheromone along its path, so that the following ants will be likely to follow the same path. The pheromone has a evaporation rate, so if the path is not efficient, the pheromone will completely vanish and the following ants will try a different path. There are many variants of the ACO [12]. In this application it has been used the Ant System as described in [13].

At each iteration, each ant chooses a path from the starting point to the goal point along a graph, and the ants can move only from one layer to the next one. The path is chosen by combining an heuristic, the pheromone deposited on the graph and a casual component:

$$\begin{aligned} P &= heuristic \cdot pheromon \\ P_{norm} &= \frac{P}{\sum P} \\ node &= find\Big(rand[0,1] \leq sum\ cum(P_{norm}), closest\Big) \end{aligned} \tag{B.1}$$

The pheromone deposition is performed at the end of every iteration and is inverse-proportional to a cost function. In Algorithm 3 is shown the pseudo-code of the ACO implemented.