Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Inverse Kinematics of Serial Manipulators in Cluttered Environments using a new Paradigm of Particle Swarm Optimization

Riccardo Falconi, Raffaele Grandi, Claudio Melchiorri

e-mail: {riccardo.falconi, raffaele.grandi, claudio.melchiorri}@unibo.it

Abstract: In this paper, a new Behavioral-based Particle Swarm Optimization algorithm is proposed in order to solve the inverse kinematics problem for a manipulator operating in an environment cluttered with obstacles. The introduced variant of the Particle Swarm Optimization relies on the idea of dividing the population of the particles in subgroups, each of which with a specific task, achieving in this manner a faster convergence to the final result. The proposed algorithm is exploited in order to solve the inverse kinematics problem for a generic serial manipulator both in its *dexterous* and *reachable* workspace.

Keywords: Particle swarm optimization, Inverse kinematics, Robotic manipulator, Obstacle avoidance

1. INTRODUCTION

The solution of the inverse kinematics (IK) for a generic manipulator has an important role in robotics, being the task assigned to a robot manipulator typically defined in the Cartesian space, while the robot itself is usually controlled in the joint space. As well known, given a joint configuration, the forward kinematics for serial manipulators has always one solution, while the IK problem may have one, none, or multiple solutions, (Siciliano and Khatib, 2008). Moreover, in the solution of the IK problem in general one has to deal also with singularities and workspace limits. Except for some well-known families of industrial robots, in general there is no closed-form solution for the inverse kinematics problem of a serial manipulator. For this reason, this problem has been widely studied in the last decade. One of the most exploited algorithms for the solution of the inverse kinematics for a generic serial manipulator was developed by Siciliano (1990). In this work, the author proposed the so-called Closed Loop Inverse Kinematics (CLIK) algorithm, an iterative method to compute the joint configuration corresponding to a desired final pose (position and orientation) of the end-effector. This approach is based on the pseudo-inverse of the Jacobian matrix J (or its transpose), from which it follows the high computational complexity of the algorithm. Moreover, it cannot take into account the presence of obstacles in the robot's workspace.

For these reasons, in recent years many other techniques have been proposed to solve the IK of a manipulator. For example, Beheshti and Tehrani (1999) implemented an adaptive fuzzy logic controller in order to calculate a suitable joint configuration for a desired pose of the end-effector while avoiding obstacles. Another interesting approach, used for example by Dash et al. (2011), is based on the implementation of neural networks. The main drawback of this approach, anyway, is the necessity of properly train the network in order to reach a sufficiently accurate solution. Another proposed solutions for the IK problem have been developed based on the Genetic Algorithms (GAs). As an example, in (Nearchou, 1998) a two-level genetic algorithm is exploited to estimate the a suitable configuration of the joint variables for the robot operating in complex environments.

One of the most promising techniques for the solution of non-linear constrained optimization problems is the Particle Swarm Optimization (PSO) algorithm. This optimization technique has been introduced for the first time by Kennedy and Eberhart (1995) and has been exploited in a wide range of practical applications.In particular, PSO and its variants have been applied also to the solution of the IK for serial manipulators.For example, Du and Wu (2011) implemented an improved PSO to estimate the IK of reconfigurable modular robots, while in (Huang et al., 2012) the PSO is applied to a 7-DOF manipulator. In spite of the many techniques proposed to solve the IK problem, rarely the presence of constraints is considered. The aim of this work is to introduce a new paradigm of PSO algorithm to compute a solution of the IK of generic serial manipulators, considering constraints in both the joint and in the work space.

This paper is organized as follows: in Sec. 2 the classic PSO is introduced and the proposed variant of the algorithm is elucidated. In Sec. 3 the Inverse Kinematic problem is introduced and an appropriate *fitness function* is defined. In Sec. 4 the simulation results used to validate our approach are presented and discussed, while Sec. 5 concludes with final remarks.

⋆ Riccardo Falconi, Raffaele Grandi and Claudio Melchiorri are with DEI, the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, Italy.

2. AN IMPROVED PARTICLE SWARM OPTIMIZATION ALGORITHM

2.1 Background on Particle Swarm Optimization

In the original formulation of the PSO algorithm (Kennedy and Eberhart, 1995), the authors presented an optimization algorithm that consists of a number of particles moving in the search space of the given optimization problem. More specifically, given a *fitness function* $f(\cdot)$ depending on M parameters, the position of each particle represents a possible solution. The position and velocity of the i-th particle of the population $\mathcal{P}$ at the k-th iteration are defined by the M-dimensional vectors

$$\boldsymbol{q}^i(k) = \left[q_1^i(k) \dots q_M^i(k)\right]^T \tag{1}$$

$$\boldsymbol{v}^i(k) = \left[v_1^i(k) \dots v_M^i(k)\right]^T \tag{2}$$

for $i = 1 \dots \|\mathcal{P}\|$, where the operator $\|\cdot\|$ represents the cardinality of a given set. By assuming a communication topology between the particles of $\mathcal{P}$, the *Neighbors subset* w.r.t the i-th particle is defined as

$$\mathcal{N}_i(k) = \{j \in \mathcal{P} : i \Leftrightarrow j\}$$

where the symbol $\Leftrightarrow$ is used to address the exchange of information between particles. Let us remark that, as pointed out by Akat and Gazi (2008) , the communication topology can affect the speed of convergence of the PSO algorithm.

In the PSO algorithm, the trajectory of each particle through the search space depends on few key information, in particular on the *personal best solution* $\boldsymbol{q}_p^i(k)$, the *neighbors best* $\boldsymbol{q}_n^i(k)$ and the *global best* $\boldsymbol{q}_g(k)$. These three terms correspond respectively to the coordinates of the best value the i-th particle found so far, the best value found by the neighbors $\mathcal{N}_i$ and the best value found by the whole population $\mathcal{P}$. More specifically, by assuming we are considering the minimization problem of a fitness function $f(\cdot)$, the terms $\boldsymbol{q}_n^i(k)$ and $\boldsymbol{q}_g(k)$ are defined respectively as

$$\boldsymbol{q}_n^i(k) = \left\{\boldsymbol{q}_p^j(k) \text{ s.t. } f(\boldsymbol{q}_p^j(k)) < f(\boldsymbol{q}_p^h(k)),\ \forall h \in \mathcal{N}_i(k)\right\}$$

$$\boldsymbol{q}_g(k) = \left\{\boldsymbol{q}_p^j(k) \text{ s.t. } f(\boldsymbol{q}_p^j(k)) < f(\boldsymbol{q}_p^h(k)),\ \forall h \in \mathcal{P}\right\}$$

As the value of $\boldsymbol{q}_g(k)$ is the same for all the particles, the exponent i in the left term of the previous equation has been omitted.

At its start, the PSO algorithm initializes the particles positions $\boldsymbol{q}^i(0)$ and velocities $\boldsymbol{v}^i(0)$ with random values, typically limited in a predefined range. At each iteration, the velocity and the position of the i-th particle are updated according to the following equations

$$\boldsymbol{v}^i(k+1) = {}^0\chi \left(w^i \cdot \boldsymbol{v}^i(k) + \phi_p \cdot \boldsymbol{\Delta}_p^i(k) + \right.$$
$$\left. + \phi_n \cdot \boldsymbol{\Delta}_n^i(k) + \phi_g \cdot \boldsymbol{\Delta}_g^i(k) \right) \tag{3}$$

$$\boldsymbol{q}^i(k+1) = \boldsymbol{q}^i(k) + \boldsymbol{v}^i(k+1) \tag{4}$$

where w^i is a scalar constant that represents the *inertia* of the i-th particle, ${}^0\chi$ is a random number in [0.9, 1] called *constriction factor* (Eberhart and Shi, 2000) introduced to limit the speed of the particles, and ϕ_p, ϕ_n, ϕ_g are tuning parameters modulating the influence that each component has on the velocity update. The values of theses parameters are typically tuned empirically (e.g. see (Tewolde et al., 2009)).The terms $\boldsymbol{\Delta}_p^i(k)$, $\boldsymbol{\Delta}_n^i(k)$ and $\boldsymbol{\Delta}_g(k)$ represent the velocity contribution given by the *best personal* $\boldsymbol{q}_p^i(k)$, the *best neighborhood* $\boldsymbol{q}_n^i(k)$ and the *best global* $\boldsymbol{q}_g$, respectively. They are defined as

$$\begin{aligned} \boldsymbol{\Delta}_p^i(k) &= \lambda_p \cdot \left(\boldsymbol{q}_p^i(k) - \boldsymbol{q}^i(k)\right) \\ \boldsymbol{\Delta}_n^i(k) &= \lambda_n \cdot \left(\boldsymbol{q}_n^i(k) - \boldsymbol{q}^i(k)\right) \\ \boldsymbol{\Delta}_g^i(k) &= \lambda_g \cdot \left(\boldsymbol{q}_g(k) - \boldsymbol{q}^i(k)\right) \end{aligned} \tag{5}$$

where λ_p, λ_n and λ_g are random numbers from the standard uniform distribution on the open interval $]0, 1[$. They are computed at each iteration of the algorithm in order to give a certain degree of variation to the behavior of the particles.

2.2 Behavioral-based Particle Swarm Optimization

Despite its success in real world applications, the PSO algorithm has some drawbacks. In fact, the trajectories of the particles and their efficiency in exploring the search space depend on a set of constants that have to be carefully tuned case by case. Moreover, depending on the initial position of the particles, they can be attracted too fast to a local minimum region (*plateau*) and thus they could not be able to properly explore the search space. To avoid these limitations, we have defined a new paradigm of PSO. In our approach, the particles of $\mathcal{P}$ are divided in S distinct subgroups, each one with a different behavior in order to improve the performances of the algorithm. More formally:

$$\bigcup_{h=1}^{S} \mathcal{G}_h = \mathcal{P} \quad \wedge \quad \bigcap_{h=1}^{S} \mathcal{G}_h = 0$$

In our modified version of the PSO algorithm, the information exchange between particles is assured by a *fully informed* communication topology, i.e. each particle is virtually connected to any other one.

For the sake of clarity, let us consider the case of three different subgroups (i.e. $S = 3$) optimizing the generic fitness function $f(\cdot)$. The behavior of the particles in each subgroup has been designed in order to globally speed up the convergence of the algorithm and, at the same time increase the explored area of the search space (i.e. avoiding the particles to be trapped in a local plateau).

The particles in $\mathcal{G}_1$ perform *local optimization* by exploiting the gradient descend algorithm (Snyman, 2005), thus their velocities are updated at each iteration according to

$$\boldsymbol{v}^i(k+1) = -{}^1w \cdot \nabla f(\boldsymbol{q}^i(k)) - {}^1k_g \cdot \boldsymbol{\Delta}_g^i(k), \quad \text{for } i \in \mathcal{G}_1 \tag{6}$$

where ${}^1w$ is the inertia characterizing group $\mathcal{G}_1$ and ${}^1k_g$ is a tuning constant. The term ${}^1k_g \cdot \boldsymbol{\Delta}_g^i(k)$ has been introduced in order to assure that, if the *global* best $\boldsymbol{q}_g(k)$, is away from the particles in $\mathcal{G}_1$ they can escape local mimina. The particles of $\mathcal{G}_2$ and $\mathcal{G}_3$ update their velocities by exploiting a modified version of the PSO algorithm, namely:

$$\boldsymbol{v}^i(k+1) = {}^h\chi \left({}^hw \cdot \boldsymbol{v}^i(k) + {}^h\gamma_p(k) \cdot \boldsymbol{\Delta}_p^i(k) + \right.$$
$$\left. + {}^h\gamma_n(k) \cdot \boldsymbol{\Delta}_n^i(k) + {}^h\gamma_g(k) \cdot \boldsymbol{\Delta}_g^i(k) \right) \tag{7}$$

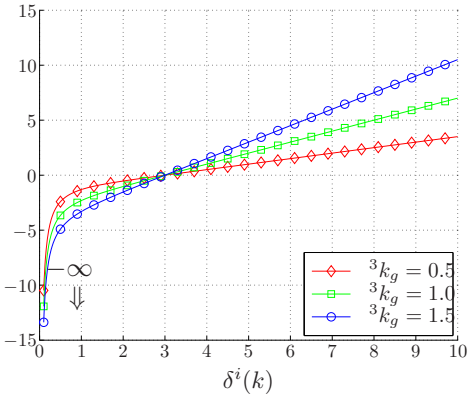

Fig. 1. Function $^3\gamma_g(k)$ depending on the value of $\delta^i(k)$ for different values of ^3k_g (here $\Omega(k) = 3$).

for $i \in \mathcal{G}_h$. The terms $^h\chi$, ^hw are respectively the constriction factor and the inertia of the particles of the h-th group, while the terms $\boldsymbol{\Delta}_p^i(k)$, $\boldsymbol{\Delta}_n^i(k)$ and $\boldsymbol{\Delta}_g^i(k)$ are the same as defined in Eq. 5. The terms $^h\gamma_p(k)$, $^h\gamma_n(k)$ and $^h\gamma_g(k)$ represent the key point of the B-PSO. In fact, depending on their definition it is possible to obtain different behaviors for the particles of each subgroup.

Before characterizing in details the behavior of subgroups $\mathcal{G}_2$ and $\mathcal{G}_3$, let us introduce the *memory factor* $\zeta(k)$ (Grandi et al., 2012) defined as

$$\zeta(k+1) = \begin{cases} \zeta(k)+1 & \text{if } \tilde{\boldsymbol{q}}_g(k+1) \leq \tilde{\zeta} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

where $\tilde{\boldsymbol{q}}_g(k+1) = |\boldsymbol{q}_g(k+1) - \boldsymbol{q}_g(k)|$ and the value of $\tilde{\zeta}$ is a constant threshold that represents how far $\boldsymbol{q}_g(k+1)$ has to be from $\boldsymbol{q}_g(k)$ to be considered a *new* optimal solution. Roughly speaking, the value of $\zeta(k)$ increases as long as the value of $\boldsymbol{q}_g(k+1) \approx \boldsymbol{q}_g(k)$, i.e. the algorithm does not find a *significantly* better optimal solution.

It is now possible to define the behaviors of the particles in $\mathcal{G}_2$ and $\mathcal{G}_3$ by properly define the terms $^h\gamma_p(k)$, $^h\gamma_n(k)$, $^h\gamma_g(k)$. More specifically, the particles in group $\mathcal{G}_2$ are used as *short-range* explorers, i.e. they search for optimal solutions in an area of the search space close to the actual best solution $\boldsymbol{q}_g(k)$. To define their behavior, the parameters introduce in the Eq. 7 (for $h = 2$) are:

$$\begin{aligned} ^2\gamma_p(k) &= {}^2k_p \cdot \zeta^{-1}(k) \\ ^2\gamma_n(k) &= {}^2k_n \cdot \zeta^{-1}(k) \\ ^2\gamma_g(k) &= {}^2k_g \cdot \zeta(k) \end{aligned} \tag{9}$$

where the constants ^2k_p, ^2k_n, ^2k_g are tuning parameters typically set to 2.05 (Hu and Eberhart, 2002). From Eq. 8 and Eq. 9, it follows that as long as the best solution is not substantially improved, and thus the value of $\zeta(k)$ increases, the influence of the terms $^2\gamma_p(k)$ and $^2\gamma_n(k)$ vanishes and the effect of $^2\gamma_g(k)$ increases. Moreover, as $0.9 \leq \chi^i \leq 1$, the particles in $\mathcal{G}_2$ tend to collapse on $\boldsymbol{q}_g$. In this way, $\mathcal{G}_2$ explores the area surrounding $\boldsymbol{q}_g(k)$.

The particles in group $\mathcal{G}_3$ are used instead for *long-range* exploration, i.e. their task is to search possible optimal solution in region of the search space that are far away from the actual $\boldsymbol{q}_g(k)$. In this case, the parameters $^h\gamma_p(k)$, $^h\gamma_n(k)$, $^h\gamma_g(k)$, (for $h = 3$) are given by

$$\begin{aligned} ^3\gamma_p(k) &= 0 \\ ^3\gamma_n(k) &= 0 \\ ^3\gamma_g(k) &= {}^3k_g \cdot \left(\delta^i(k) - \frac{1}{\tanh(\delta^i(k))} + \Omega(k) \right) \end{aligned} \tag{10}$$

where ^3k_g is a constant typically set to 2.05 (Hu and Eberhart, 2002) and $\delta^i(k) = |\boldsymbol{q}_g(k) - \boldsymbol{q}^i(k)|$ is the Euclidean distance between the i-th particle and the actual $\boldsymbol{q}_g^i(k)$. The term $\Omega(k)$ in Eq. 10 is a function of a threshold parameter $\rho(k)$, namely

$$\Omega(k) = \frac{1}{\tanh(\rho(k))} - \rho(k) \tag{11}$$

i.e. if distance between the i-th particle and $\boldsymbol{q}_g(k)$ is lower than a threshold $\rho(k)$, then the area around $\boldsymbol{q}_q(k)$ is repulsive, attractive otherwise (see Fig. 1). In order to ensure that the particles in $\mathcal{G}_3$ can act as *long-range* explorers able to adapt their behavior depending on the *fitness function*, the value of $\rho(k)$ is defined as a function of the *memory factor* as

$$\rho(k) = \alpha \cdot |\sin(\beta \cdot \zeta(k))| \tag{12}$$

where the two parameters α, β are used to define the maximum radius of expansion of the repulsive zone and the frequency in the variation of $\rho(k)$, respectively.

The choice of the values in Eq. 10 can be explained as follows. From Eq. 10 and Eq. 11, the increasing value of $\zeta(k)$ leads to a value of $\Omega(k)$ that changes according to Eq. 12, thus pushing the particles in $\mathcal{G}_3$ away from $\boldsymbol{q}_g(k)$. As a consequence, as long as the best solution is not consistently improved, the group $\mathcal{G}_3$ is pushed to explore areas of the search space that are far from $\boldsymbol{q}_g$. Once $\boldsymbol{v}^i(k+1)$ is calculated, the position of each particle is then updated as in Eq. 4.

The effectiveness of the B-PSO algorithm w.r.t. the classic PSO algorithm has been investigated by collecting statistical results gathered with Matlab simulations. In particular, a set of benchmark functions typically used in the literature to test meta-heuristic optimization algorithms has been considered. The results are presented in (Falconi et al., 2013).

3. B-PSO APPLIED TO THE INVERSE KINEMATICS PROBLEM

The B-PSO algorithm presented in Sec. 2.2 has been exploited to solve the inverse kinematics problem for a generic M degrees of freedom manipulator operating in presence of N_o environmental obstacles.

3.1 The fitness function

The idea beyond the use of B-PSO algorithm is to calculate the inverse kinematics of a manipulator operating in an environment cluttered with obstacles by working directly in the joint space, thus avoiding any problem related to singularities. Given a M degrees-of-freedom manipulator, each particle of $\mathcal{P}$ represents a possible joint configuration of the manipulator.

The direct kinematics function of the robot whose joint configuration is defined by $\boldsymbol{q}^i(k)$ is thus given by

$$f_{kine}(\boldsymbol{q}^i(k)) = {}^0\boldsymbol{T}_M(\boldsymbol{q}^i(k)) = \begin{bmatrix} {}^0\boldsymbol{R}^i_M(k) & {}^0\boldsymbol{p}^i_M(k) \\ [0\ 0\ 0] & 1 \end{bmatrix} \quad (13)$$

where ${}^0\boldsymbol{R}^i_M(k) \in \mathbb{R}^{3\times3}$, ${}^0\boldsymbol{p}^i_M(k) \in \mathbb{R}^{3\times1}$ represent the orientation and position of the end-effector depending on the current joint configuration, respectively. The desired final pose of the end-effector can be defined by the homogeneous matrix $\boldsymbol{T}_d \in \mathbb{R}^{4\times4}$, i.e.

$$\boldsymbol{T}_d = \begin{bmatrix} \boldsymbol{R}_d & \boldsymbol{p}_d \\ [0\ 0\ 0] & 1 \end{bmatrix} \quad (14)$$

where $\boldsymbol{R}_d \in \mathbb{R}^{3\times3}$ and $\boldsymbol{p}_d \in \mathbb{R}^{3\times1}$ represent the desired orientation and position of the end-effector, respectively. From Eq. 13 and Eq. 14, it is thus possible to calculate at instant k the position error $\Delta_p(k)$ and the angular error $\Delta_a(k)$. More formally, the position error is defined as

$$\Delta_p(k) = \left(\boldsymbol{p}_d - {}^0\boldsymbol{p}^i_M(k)\right)^2 \quad (15)$$

To calculate the angular error and at the same time avoid the singularities that typically arise when the Euler angles are used, we have used the quaternion representation. As it is well known, a unit quaternion expressed in a given frame F_w is a four dimensional vector $\boldsymbol{\epsilon} = \left[\epsilon_0, \boldsymbol{\eta}^T\right]^T \in \mathbb{R}^4$, where ϵ_0 denotes the scalar part of the quaternion and $\boldsymbol{\eta} = [\epsilon_1,\ \epsilon_2,\ \epsilon_3]^T$ denotes its vectorial part. Moreover

$$\boldsymbol{\epsilon} = \epsilon_0 + \epsilon_1\hat{i} + \epsilon_2\hat{j} + \epsilon_3\hat{k}$$

with the constraint $\epsilon_0^2+\epsilon_1^2+\epsilon_2^2+\epsilon_3^2 = 1$; $\hat{i}, \hat{j}, \hat{k}$ are the unit vectors of the x, y, z axes of frame F_w. Given a rotation matrix $\boldsymbol{R} = [R_{ij}] \in \mathbb{R}^{3\times3}$ that describes the asset of the body frame F_b with respect to the inertial frame F_w, the corresponding quaternion can be computed as:

$$\begin{aligned} \epsilon_0 &= \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}}, & \epsilon_1 &= \frac{R_{32} - R_{23}}{4 \cdot \epsilon_0} \\ \epsilon_2 &= \frac{R_{13} - R_{31}}{4 \cdot \epsilon_0}, & \epsilon_3 &= \frac{R_{21} - R_{12}}{4 \cdot \epsilon_0} \end{aligned} \quad (16)$$

The derivative of the quaternion vector is calculated using the *quaternion propagation rule*, namely $\dot{\boldsymbol{\epsilon}} = \left[\dot{\epsilon}_0, \dot{\boldsymbol{\eta}}^T\right]^T$ with

$$\dot{\epsilon}_0 = -\frac{1}{2}\hat{\boldsymbol{\epsilon}}^T\boldsymbol{\omega} \qquad \dot{\boldsymbol{\eta}} = \frac{1}{2}\left[\epsilon_0\boldsymbol{I}_3 + \boldsymbol{\eta}^\times\right]\boldsymbol{\omega} \quad (17)$$

being $\boldsymbol{I}_3$ the 3×3 identity matrix. Further discussions on quaternion properties and on the quaternion propagation rule can be found e.g. in (Caccavale and Villani, 1999).Given two generic quaternions $\boldsymbol{\epsilon}_1 = \left[a_0,\ \boldsymbol{\eta}_a^T\right]^T$ and $\boldsymbol{\epsilon}_2 = \left[b_0,\ \boldsymbol{\eta}_b^T\right]^T$, the angular error $\Delta_a(\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2) = \left[\delta\epsilon_0,\ \delta\boldsymbol{\eta}^T\right]^T$ is defined as:

$$\Delta_a(\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2) = \begin{bmatrix} \delta\epsilon_0 \\ \delta\boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} a_0 & -\boldsymbol{\eta}_a^T \\ \boldsymbol{\eta}_a & a_0\boldsymbol{I}_3 + \boldsymbol{\eta}_a^\times \end{bmatrix} \begin{bmatrix} b_0 \\ -\boldsymbol{\eta}_b \end{bmatrix} \quad (18)$$

where $\boldsymbol{\eta}_a^\times$ is the 3×3 *skew symmetric* matrix generated from the vector $\boldsymbol{\eta}_a$, i.e.

$$\boldsymbol{\eta}_a = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \Longrightarrow \boldsymbol{\eta}_a^\times = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}.$$

By exploiting Eq. 18, it is now possible to define the angular error between the desired and the actual orientation of the manipulator's end-effector, i.e.

$$\Delta_a(k) = \Delta_a(\boldsymbol{\epsilon}_d, \boldsymbol{\epsilon}^i(k))^2 \quad (19)$$

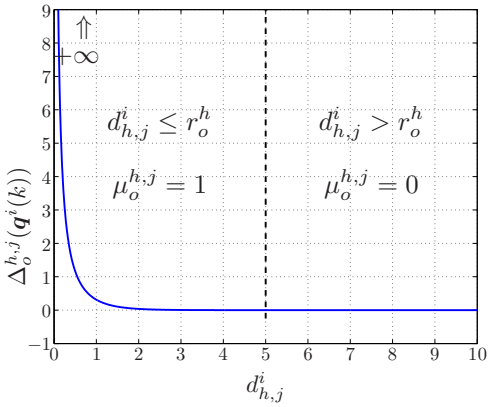

Fig. 2. Profile of the function in Eq. 21, assuming spherical obstacles.

where $\boldsymbol{\epsilon}^i(k)$ and $\boldsymbol{\epsilon}_d$ are the quaternions corresponding to ${}^0\boldsymbol{R}^i_M(k)$ and $\boldsymbol{R}_d$ as introduced in Eq. 13 and Eq. 14, respectively.

In order to include the N_o obstacles placed in the robot's workspace as part of the fitness function, another term $\Delta_o(k)$ has to be defined. Namely,

$$\Delta_o(k) = \sum_{h=1}^{N_o}\sum_{j=1}^{M}\Delta_o^{h,j}(\boldsymbol{q}^i(k)) \quad (20)$$

where $\Delta_o^{h,j}(\boldsymbol{q}^i(k))$ takes into account the *fitness cost* corresponding to a collision between the h-th obstacle and the j-th link of the manipulator. Roughly speaking, the term $\Delta_o(k)$ maps the obstacles from the robot's workspace to its joint space.

For the sake of clarity, let us consider obstacles with spherical shape, i.e. the generic h-th obstacle is fully described by its radius r_o^h and the position of its center $\boldsymbol{p}_o^h$ w.r.t. a common reference frame F_w (typically the robot's base frame). Furthermore, let us suppose that the links of the manipulator can be represented as segments.

Then, the term $\Delta_o^{h,j}(\boldsymbol{q}^i(k))$ introduced in Eq. 20 is defined as (see Fig. 2),

$$\Delta_o^{h,j}(\boldsymbol{q}^i(k)) = \mu_o^{h,j}\left(\tanh\left(\frac{1}{d^i_{h,j}}\right) - \tanh\left(\frac{1}{r_o^h}\right)\right) \quad (21)$$

where $d^i_{h,j}$ is the Euclidean minimum distance between the center of the h-th obstacle and the j-th link of the manipulator (see Fig. 3). The value of $\mu_o^{h,j}$ is a constant such that

$$\mu_o^{h,j} = \begin{cases} 1 & \text{if } d^i_{h,j} \leq r_o^h \\ 0 & \text{otherwise} \end{cases}$$

Finally, the fitness function exploited by B-PSO is defined as

$$f(\boldsymbol{q}^i) = K_p \cdot \Delta_p(k) + K_a \cdot \Delta_a(k) + K_o \cdot \Delta_o(k) \quad (22)$$

where $K_p \geq 0$, $K_a \geq 0$ and $K_o \geq 0$ are a tuning parameters.

Let us remark that, depending on the value of the tuning parameters K_p, K_a in Eq. 22, the B-PSO algorithm can solve the inverse kinematics function w.r.t. the robot *dextrous* workspace ($K_p > 0 \wedge K_a > 0$) or w.r.t. its *reachable* workspace ($K_p > 0 \wedge K_a = 0$) (Siciliano and Khatib, 2008).

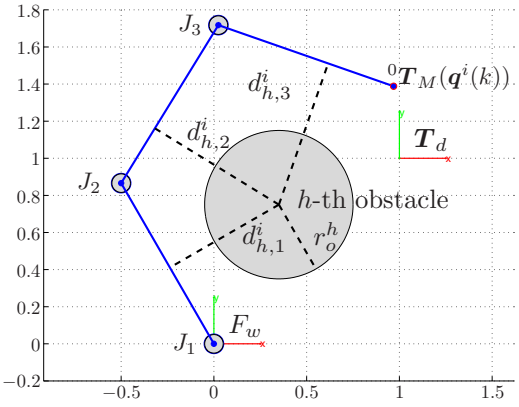

Fig. 3. An example of a planar manipulator performing a positioning task in an environment with circular obstacles.

Table 1. Denavit-Hartenber parameters of the 3-DOF planar manipulator

	d_i	θ_i	a_i	α_i
Joint 1	0	θ_1	1	0
Joint 2	0	θ_2	1	0
Joint 3	0	θ_2	1	0

4. SIMULATIONS

In order to prove the effectiveness of the presented algorithm, several Matlab simulations considering different scenarios have been performed. In the following, without loss of generality, we refer to the planar manipulator with three rotational joints described by the Denavit-Hartenberg parameters reported in Tab. 1. As a consequence, each particle performing B-PSO is now defined as $\boldsymbol{q}^i(k) = [\theta_1,\ \theta_2,\ \theta_3]^T$, where θ_1, θ_2, θ_3 are the joint variables. More in details, two different cases have been considered in our simulations. In the first case, we have studied an environment with two obstacles (see Fig. 4(a)) and the IK problem has been solved in the *dexterous workspace*, i.e. in the fitness function defined in Eq. 22 we set $K_p = K_a = K_o = 1$. In the second case, we have tested two scenarios with increasing complexity (see Fig. 4(b) and Fig. 4(c), solving the inverse kinematics problem w.r.t. the *reachable* workspace of the manipulator, i.e. in Eq. 22 we set $K_p = K_o = 1$ and $K_a = 0$.

Data have been collected by running 100 simulations on the three different scenarios (see Fig. 4). Similar simulations have been performed also with the standard PSO algorithm introduced in Sec. 2.1, thus providing a significant comparison between the two methodologies. Let us remark that it is not possible to compare the B-PSO with the CLIK algorithm since this cannot deal with obstacles placed in the robot's workspace. More in particular, both the PSO and B-PSO algorithms exploit a population of a total of 20 particles that, in the case of B-PSO with three subgroups, are divided as $\|\mathcal{G}_1\| = 1$, $\|\mathcal{G}_2\| = 5$, $\|\mathcal{G}_3\| = 14$ (or, for brevity, $1 - 5 - 14$).

An analysis of the simulation data regarding the performances of PSO and B-PSO is reported in Tab. 2, where for both the algorithms the mean convergence time (T_{PSO}, T_{B-PSO}), the time standard deviation ($\hat{T}_{PSO}$, $\hat{T}_{B-PSO}$) and the success rate ($S_{\%,PSO}$, $S_{\%,B-PSO}$) are reported. The time efficiency η_T of the B-PSO w.r.t. the classic PSO algorithm is reported in the last column, where

$$\eta_T = \frac{T_{PSO}}{T_{B-PSO}}$$

Each simulation is concluded if the algorithm reaches a configuration corresponding to a value of the fitness function lower than a predefined threshold ε or if the number of iteration is greater than a predefined value k_{max}, i.e. the algorithm stops when

$$\left(f(\boldsymbol{q}_g(k)) \leq \varepsilon \right) \vee \left(k > k_{max} \right)$$

In our experiments, we set $\varepsilon = 10^{-3}$ and $k_{max} = 1000$.

4.1 Discussion of the results

As it can be seen in Tab. 2, the B-PSO algorithm presents better performances w.r.t. the standard PSO. In each scenario the PSO has a success rate that is significantly low, while the B-PSO algorithm is always able to find a solution for the IK problem. Moreover, even in case of success, the PSO presents a significantly higher convergence time if compared with B-PSO. The low convergence rate of the PSO is related to the fact that the obstacles introduced in the workspace are mapped into the joint space through $\Delta_o(k)$ (in Eq. 20), thus the search space presents many plateaus where the swarm can eventually be trapped. In fact, while on one side the $^0\chi$ factor introduced in Eq. 3 avoids the swarm to diverge, on the other side it forces the particles to collapse to $\boldsymbol{q}_g(k)$. As a consequence, after few iterations the swarm is not able to properly explore the search space. This problem has been avoided in B-PSO by introducing the *long-range* explorers (particles in group $\mathcal{G}_3$). On the other side, the particles in $\mathcal{G}_1$ and $\mathcal{G}_2$ have a behavior that leads them to local optimization in an area of the search space localized around the current $\boldsymbol{q}_g(k)$, thus speeding up the convergence of the algorithm. It is worth to notice that, as it could be expected, by increasing the complexity of the environment, the average number of iterations required to reach $f(\boldsymbol{q}_g(k)) \leq \varepsilon$ increases. At the same time, the success rate of the PSO algorithm decreases while the success rate of the B-PSO is always 100%.

5. CONCLUSIONS

In this paper, a Behavioral-based Particle Swarm Optimization algorithm has been presented in order to solve the inverse kinematics problem for a serial robot operating in an environment cluttered with obstacles. The defined fitness function can take into account not only the problem of positioning the end-effector in a desired point in the space, but it considers also the problem of its final orientation. The effectiveness of the B-PSO regarding the IK problem have been proved by means of simulations. By comparing the gathered results with analogous simulations performed with standard PSO algorithm, it follows that not only our approach has a 100% success rate, but also that the B-PSO converges faster than PSO.

REFERENCES

Akat, S.B. and Gazi, V. (2008). Particle swarm optimization with dynamic neighborhood topology: Three

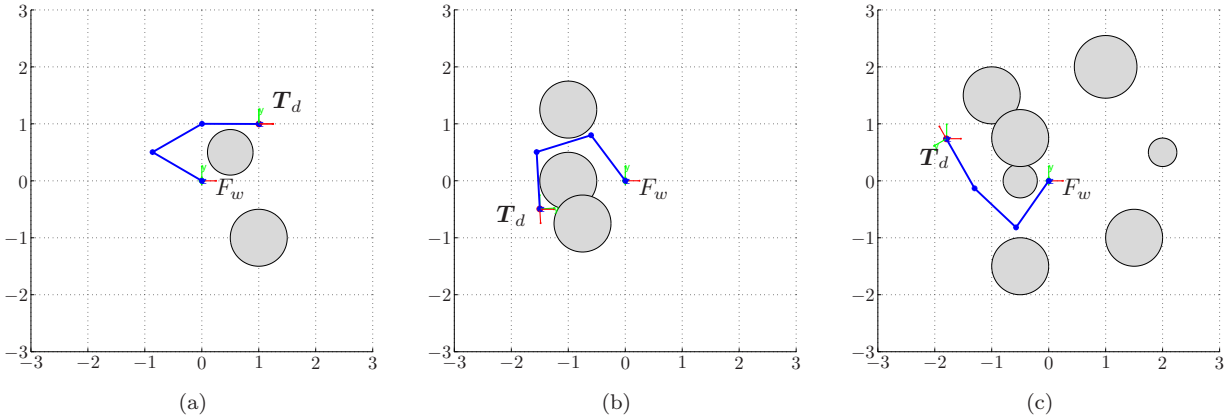

Fig. 4. Scenarios with increasing complexity considered in the simulation runs. One of the solutions calculated by B-PSO is depicted in each scenario.

Table 2. Results collected over 100 simulation run w.r.t. the environments reported in Fig. 4.

	PSO with N=20			B-PSO with N=1-5-14			
	Mean time T_{PSO}	Standard dev. $\hat{T}_{PSO}$	Success rate $S_{\%,PSO}$	Mean time T_{B-PSO}	Standard dev. $\hat{T}_{B-PSO}$	Success rate $S_{\%,B-PSO}$	Time eff. η_T
Scenario 1 (Fig.4(a)) $K_p = K_a = K_o = 1$	85.5	215.6	85%	46.4	21.6	100%	1.84
Scenario 2 (Fig.4(b)) $K_p = K_o = 1,\ K_a = 0$	445.2	419.5	65%	67.2	19.8	100%	6.25
Scenario 3 (Fig.4(c)) $K_p = K_o = 1,\ K_a = 0$	628.1	393.1	50%	78.1	25.3	100%	8.04

neighborhood strategies and preliminary results. *2008 IEEE Swarm Intelligence Symposium*, 1–8.

Beheshti, M. and Tehrani, A. (1999). Obstacle avoidance for kinematically redundant robots using an adaptive fuzzy logic algorithm. In *American Control Conference, 1999. Proceedings of the 1999.*

Caccavale, F. and Villani, L. (1999). Output feedback control for attitude tracking. *Systems & Control Letters*, 38(2), 91 – 98.

Dash, K.K., Choudhury, B., Khuntia, A.K., and Biswal, B. (2011). A neural network based inverse kinematic problem. In *Recent Advances in Intelligent Computational Systems (RAICS), 2011 IEEE*, 471–476.

Du, Y. and Wu, Y. (2011). Application of ipso algorithm to inverse kinematics solution of reconfigurable modular robots. In *Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on.*

Eberhart, R. and Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proc. 2000 Congress on*, 84 –88 vol.1.

Falconi, R., Grandi, R., and Melchiorri, C. (2013). A behavioral-based approach to particle swarm optimization. In *to the 10th IEEE International Conference on Robotics and Biomimetics, 2013. ROBIO 2013.*

Grandi, R., Falconi, R., and Melchiorri, C. (2012). A navigation strategy for multi-robot systems based on particle swarm optimization techniques. In *Proceedings of the 10th IFAC Symposium on Robot Control 2012.*

Hu, X. and Eberhart, R. (2002). Adaptive particle swarm optimization: detection and response to dynamic systems. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2.

Huang, H.C., Chen, C.P., and Wang, P.R. (2012). Particle swarm optimization for solving the inverse kinematics of 7-dof robotic manipulators. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, 3105–3110.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, 1942 – 1948 vol.4.

Nearchou, A.C. (1998). Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. *Mechanism and Machine Theory*, 33(3), 273 – 292.

Siciliano, B. (1990). Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent Robotic Systems*, 3, 201–212.

Siciliano, B. and Khatib, O. (eds.) (2008). *Springer Handbook of Robotics*. Springer.

Snyman, J.A. (2005). *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Applied Optimization, Vol. 97. Springer-Verlag New York, Inc., second edition.

Tewolde, G.S., Hanna, D.M., and Haskell, R.E. (2009). Enhancing performance of pso with automatic parameter tuning technique. *2009 IEEE Swarm Intelligence Symposium*, 67–73.