Robotica (2004) volume 22, pp. 513–522. © 2004 Cambridge University Press
DOI: 10.1017/S0263574704000281 Printed in the United Kingdom

On the maximum dynamic stress search problem for robot manipulators

A. Khoukhi* and A. Ghoul†

(Received in Final Form: January 26, 2004)

SUMMARY

This paper deals with the problem of the dynamic stress optimisation for manipulator robots. The maximum dynamic stress search is a challenging problem, insofar the dynamic stresses change in a very complex manner and they are strongly influenced by the payload, internal parameters of the robot, and instantaneous values of the articular variables (positions, velocities, and accelerations). This problem is solved using a hybrid simplex-genetic algorithm. This algorithm allows solving the dilemma of exploration vs. exploitation in the search space. A simulation study shows the greater performance of the proposed approach as compared to a classical approach of the configuration space inverse sweeping.

KEYWORDS: Dynamic stress; Sweeping; Genetic programming; Simplex programming; Hybrid programming; Robot manipulator.

1. INTRODUCTION

In the first generation of robotics, robots had been automata like devices. They had to execute simple and repetitive but precise movements[1]. The second wave of robotic systems has been characterized by the development of adaptive robots to their environments by following strict control objectives in the presence of disturbances. This led to robust and adaptive control strategies[2,3]. Nowadays and future robotics is required for more complex manipulation tasks, with high levels of accuracy and autonomy[1]. Examples of such robots are assistive robotics, and intervention robotics in constrained and hazardous environments. Manipulator robots constitute an important class among this generation of robotic systems. They have to ensure, according to task requirements, highly precise positioning and orientation of their end-effector with respect to a fixed independent inertial frame.

As manipulator robots perform tasks of several kinds, prior knowledge of the maximum dynamic stress induced by the payload is fundamental to estimate the robot adaptability for a given usage or task[4–6]. Indeed, according to the task designed, some robot operation abilities are given more importance than others. For instance, we give more importance to kinematic abilities (manipulability) to a cutting manipulator, while this importance is suitable for static capacities (transmission efforts and precision) for assembly and insertion tasks.

Among sets of parameters to evaluate, we mention those related to the robot topology (i.e robot shape, succession nature of links, structure types). This problem has been recently partially solved by applying evolutionary programming[5]. Partially because in this approach, the parameterisation is extensively depending on engineer and designer knowledge. The second parameters set is that related to the robot geometry (i.e link lengths, Denavit-Hartenberg parameters, geometry of links and joints of the robot)[7–13].

The third set of parameters and to which we are interested in this paper is how to strengthen the generating of joint space commands that limit to acceptable levels the amount of dynamic stress imposed on the manipulator.

To this end, we compute torque values with respect to payload and dynamic performance parameters of the robot. As the robot will manipulate the payload in several situations, with varying articular positions, speeds and accelerations, and in order to prevent degrading the kinematic and dynamic performances, we have to consider the articular speed and acceleration limits corresponding to maximum value of the payload. These parameters are of $2 * \text{ndof}$ (number of degrees of freedom) dimensions. The torque must be defined regarding the payload parameters and resistance conditions of the mechanical structure[12,13].

Each stress at an articular level is a non-linear function of the articular variables $(q, \dot{q}, \ddot{q})$, link parameters and 10 mass parameters of the end-effector (masses, centres of mass, and inertia matrix). Hence, for a six degrees of freedom manipulator, dynamic stresses are functions of $3 * 6$ articular variables, of $5 * 10$ links parameters of the manipulator and of 10 parameters of the payload.

The overloading may appear at any of the 36 components of the six links for a six degrees of freedom manipulator. The high number of parameters to evaluate increases the processing complexity of the maximum dynamic stress search.

Similar large-scale problems had been solved using sequential optimisation methods, and both deterministic and stochastic optimisation algorithms had been developed in the literature with successful implementations on related robot control problems[14–16]. The basic features of these methods consist in decomposing the initial problem into a sequence of sub-problems, then solving each sub-problem independently. A sub-problem can use outcomes of other sub-problems. However, such approaches increase time processing, which tends to be prohibitive. There have been many oriented evolutionary programming approaches proposed in the

* 3865 # 5 rue Beaubien E, Montreal, PQ (Canada) H1X 1H3. Corresponding author. E-mail: khoukhi@sympatica.ca
† USTHB, Department of Electrical Engineering, Bab – Ezzouar Algiers (Algeria).

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

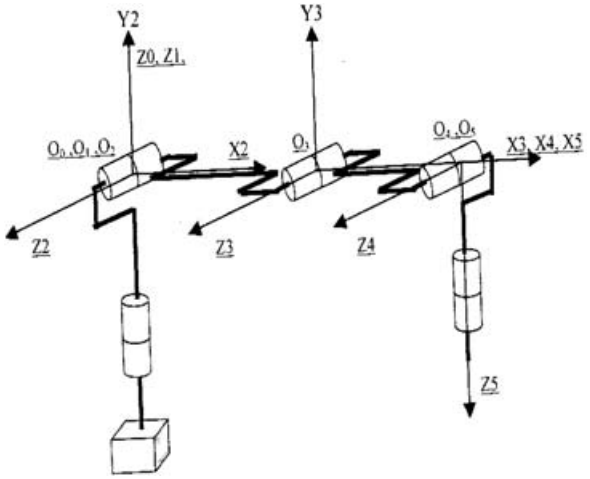


Fig. 1. Schematic representation of a STAUBLI RX-90 robot.

literature to solve large scale and complex optimisation problems[17–24].

In this paper, the genetic programming is explored to solve the stated problem by developing a hybrid simplex-genetic algorithm (SGA). We show the high efficiency of the proposed method similar to the configuration space sweeping approach that had been applied to the considered problem.

The second section develops the dynamic and maximum dynamic stress models of the robot. In the third section, we apply the sweeping approach to the considered problem. Section four gives the hybrid simplex-genetic algorithm and its application to maximum dynamic stress search. This is followed by section five, where simulation results are given, and then in section six, we draw conclusions and describe future trends.

2. MAXIMUM DYNAMIC STRESS MODELLING

This problem consists in the evaluation and finding the kinematic parameters $(q, \dot{q}, \ddot{q})$ that maximize the joint torques. Each stress at an articular level is represented by a non-linear function of a $3 * 6$ articular position, speed, and acceleration variables.

The evaluation of the Maximum Dynamic Stress (MDS) is based on the Newton-Euler dynamic model. The simulations had been achieved on the STAUBLI RX-90 robot model (Figure 1). For further details about this robot, we refer to reference[25]. The dynamic model gives the torques at the articulation levels as functions of the articular variables $(q, \dot{q}, \ddot{q})$, geometric, inertia and effort parameters of the robot.

The general statement of the dynamic stress search problem can be given as:

$$\underset{q_j, \dot{q}_j, \ddot{q}_j}{\text{Max.}} \quad \Gamma_{i,j} = |f(q, \dot{q}, \ddot{q}, F)|. \tag{1}$$

Subject to:

The articular position, speed and acceleration admissible domains are:

$$q_{j_{\min}} \leq q_j \leq q_{j_{\max}} \tag{2}$$

$$(\dot{q}_j)_{\min} \leq \dot{q}_j \leq (\dot{q}_j)_{\max} \tag{3}$$

$$(\ddot{q}_j)_{\min} \leq \ddot{q}_j \leq (\ddot{q}_j)_{\max} \tag{4}$$

where $(q_j)_{\min}$, $(q_j)_{\max}$, $(\dot{q}_j)_{\min}$, $(\dot{q}_j)_{\max}$, $(\ddot{q}_j)_{\min}$, and $(\ddot{q}_j)_{\max}$ are the minimum and maximum articular positions, speeds and accelerations, respectively, of the jth link. F represents the wrench (vector of forces and torques) $F_{i,j}$, the wrench component at joint i due to the movement of the link j.

3. SWEEPING APPROACH TO MAXIMUM DYNAMIC STRESS SEARCH

The task at hand is to solve the maximisation problem (1–4) described in section 2; which consists of finding the kinematic parameters $(q, \dot{q}, \ddot{q})$, maximizing (1) and satisfying (2), (3) and (4). Because of the largeness of the search space, smoothless of the objective function, and local range of conventional deterministic iterative-based search methods, an application of a pure deterministic search technique will fail to give a global optimum. Hence, an alternative deterministic approach to solve problem (1–4), is the systematic sweeping of the configuration space to find the best approximation to the maximum dynamic stress.

The sweeping algorithm consists to sweep each component of the configuration vector q_i of n-dimension in the bounded domain $[q_{\min}, q_{\max}]$ with a defined step size $q_{i_{step}}$. However, for ten (10) node discretizations of the articular variables $(q, \dot{q}, \ddot{q})$, the number of configurations becomes significantly large. For a six degrees of freedom manipulator, the number of possible combinations is 10^{18}. Sweeping these overall configurations is obviously not feasible and impracticable.

Nonetheless, from the dynamic model based on Lagrange formalism, any torque expression Γ can be decomposed into inertia, centrifugal, Coriolis, and gravity terms such that:

$$\Gamma = \sum A_j * \ddot{q}_j + \sum_{j=1}^{j=n-1} \sum_{i=j+1}^{i=n} B_{ij} * \dot{q}_i * \dot{q}_j + \sum_{i=1}^{i=n} C_j * q_j^2 + Q$$

where: A_j are the inertia terms, B_{ij} Coriolis terms, C_j centrifugal terms, and Q is the gravity force.

This expression shows that the torque is linear with respect to the acceleration, and non-linear regarding the velocity. Hence, the maximum dynamic stress appears only in cases where the robot works at its maximum acceleration. This fact allows for the reducing the sweeping only on articular positions and velocities. But, even with that, the total number of configurations to sweep is still huge; (for a six degrees of freedom manipulator, and for ten node discretizations of the position and velocity variables, this number is of the order of $2^6 * 10^{12}$). There are two major sweeping approaches:

3.1. Direct sweeping

The ith component of q_i is incremented with a step-size $q_{i_{step}}$ if and only if the component $i - 1$ has reached its maximum value. In this case, the first component changes

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

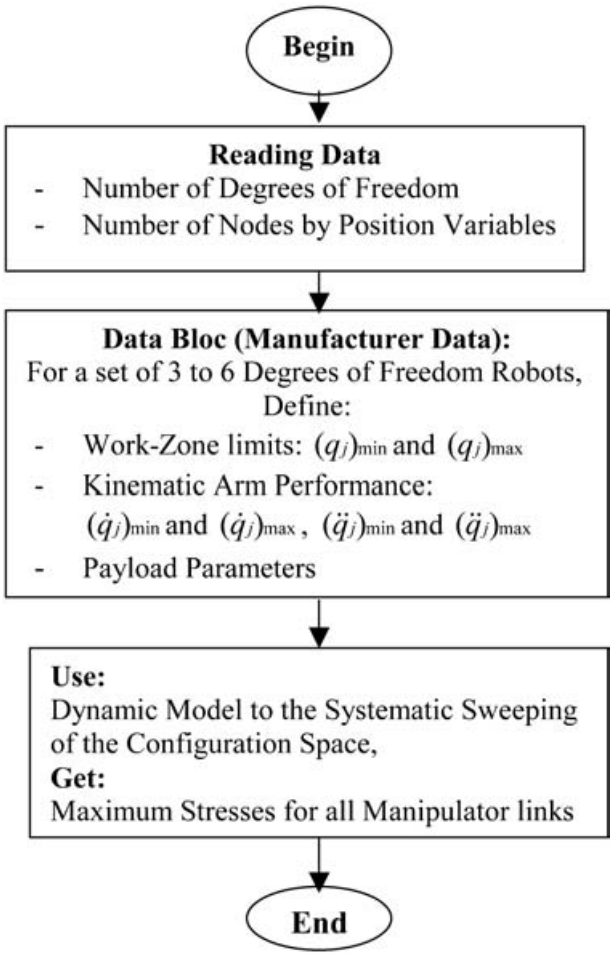


Fig. 2. Flowchart of the MDS search-based configuration space inverse sweeping using manufacturer's data.

at each iteration, and we must re-compute the Newton-Euler procedure at each iteration.

3.2. Inverse sweeping

In this approach, the ith component q_i is incremented with a step size $q_{i_{step}}$ if and only if the $i+1$th component has reached its maximum value. In this case, the last component changes at each iteration and the last but one component changes if the last one has reached its maximum, and so forth. Hence, with inverse sweeping, the computing procedure of the Newton-Euler starts with the segment corresponding to the changed component. So, we apply an inverse sweeping because of its lower time consumption.

We have developed a procedure to generate automatically different possible configurations corresponding to a given discretization of position variables of the manipulator. This procedure applies a dynamic simulation based on the Newton-Euler dynamic model of the manipulator.

Figure 2 shows a flow chart for the sweeping approach. The user can select the number of degrees of freedom of the manipulator, and the discretization steps (number of nodes).

The sweeping algorithm evaluates the components of dynamic stresses all of them simultaneously, e.g. the search space sweeping is made once to generate the $6*n$ functions (36 functions for a manipulator with n = six degrees of freedom).

4. HYBRID SIMPLEX-GENETIC ALGORITHM TO MAXIMUM DYNAMIC STRESS SEARCH

4.1. Structure of the hybrid Simplex-Genetic Algorithm (SGA)

The problem stated is a constrained multi-objective optimisation problem. It is characterized by a large search space and irregular objective function. Hence in order to explore the overall search space, we have developed a hybrid Simplex-Genetic Algorithm (SGA) that computes each component of joint torques of the robot.

To start, we define clearly the algorithm parameters. The goal is to find the state vector X of articular positions, velocities, and accelerations which represent the chromosomes or the individuals in the genetic algorithm that are solution to problem (1–4). (Throughout this paper we interchangeably use the words chromosome and individual).

We consider real-coded chromosomes, since this allows processing chromosomes with reasonable lengths, which makes more effective the actions of genetic operators. The fitness (objective) function is defined in equation (1). The initial population for the genetic search is randomly chosen with uniform distribution. The selection is made on a enlarged sampling space, where both parents and offspring have the same surviving chance. We apply an elitist selection first, then the generation is determined with a roulette wheel selection[23], which tunes each individual X_i, by:

$$E(X_i) = \frac{eval(X_i)}{\sum_{j=1}^{Pop\text{-}Size} eval(X_j)} * Pop\text{-}Size = \frac{eval(X_i)}{AvAd}$$

where $eval(X_i)$ is the ith candidate solution, *AvAd* is the average adaptation of the population, and *Pop-Size* is the population size. The rest is dispatched among candidate individuals with the proportional wheel method[23]. Thus, all individuals with adaptation above the average will certainly survive in the next generation, while all other individuals, whatever their fitness, still have a chance to be selected.

4.2. Exploitation vs. Exploration

One of strong assets of genetic algorithms is their ability to handle two conflicting goals; exploitation *versus* exploration. Throughout the search process, exploitation consists of staying as closer as possible to the best-found solution. Exploration, on the other hand, is to find new solutions in parts of the space where we have not searched yet. So, a successful optimisation algorithm has to strike a correct balance between exploitation and exploration. This constitutes a most in multi-modal optimisation-like problems, where conventional optimisation techniques fail because of multiple local optima. In genetic programming this dilemma is undertaken by the selection pressure exerted by the selection operator, and the transformation pressure exerted by mutation and crossover operators. In a reproduction process, with probabilistic rules, the selection determines

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

the conservation rate of best chromosomes, while the mutation and crossover operators control the transformation rate of the genitors. However, in conventional genetic algorithms, selection and genetic transformations (crossover and mutation) are mainly achieved at individual fitness at birth. Hybridisation of genetic algorithms by incorporating powerful local optimisation methods constitutes a significant add-on value to improve the recombination-selection loop of pure genetic algorithms. These hybrid algorithms allow genetic operators to elicit chromosomes fitness, once these chromosomes had achieved their life. The living process of each newly generated offspring is computationally made through a local optimisation iterative algorithm.

In the literature, there are several hybrid genetic algorithms that had been successfully applied to many engineering and industrial problems[23,24]. Unfortunately, the success of most of these algorithms is application-specific.

To strengthen the selection pressure and remedy the shaky balance regarding exploitation *vs.* exploration in our case study, we implement three crossover operators: The first one, with a probability P_d is the discrete crossover. The second with a probability P_c is the continuous crossover. The third is the simplex crossover, which is introduced with a probability P_s. The simplex crossover is fully inspired from hill climbing like algorithms. It acts differently from the conventional continuous and discrete crossovers. All these crossover operators are applied to the overall population of the current generation. This hybridness gives rise to the following modifications of the basic genetic algorithm:

- At each generation of the genetic algorithm, λ reproduction groups are formed; each group consists of $(n+1)$ individuals, n is the size of the chromosome.
- At each iteration, only λ descendants are generated. These λ descendants will replace the worse λ parents in the current generation.
- All crossover operators are defined over $(n+1)$individuals (rather than over two as in conventional genetic algorithms).
- The quantity $(n+1)$ comes from the fact that simplex standard rules are defined on $(n+1)$ points. The basic idea in the simplex search is to compare the values of the objective function at the $n+1$ vertices of a general simplex and move the simplex gradually toward the optimum point, according to an iterative process.

Appendix A shows the processes of discrete, continuous, and simplex crossovers.

4.3. Evolution parameters tuning of the Simplex-Genetic Algorithm

Tuning the parameters evolution consists in finding the best choice of these parameters to achieve a good function of crossover and mutation genetic operators. For a mutation operator, we used a dynamic mutation. This is designed for fine-tuning capabilities in order to achieve high-precision. This operator, also called non-uniform mutation, is introduced here briefly. For further readings, one can see, for instance, reference [23].

For a given chromosome $X = [x_1, x_2, \ldots x_k, \ldots, x_n]$ at a generation t, if the element x'_k is selected for mutation, the resulting offspring is $X = \lfloor x_1, x_2, \ldots x'_k, \ldots, x_n \rfloor$ where x'_k is randomly selected according to one of following two choices:

$$x'_k = x'_k + \left((x_{\max k} - x_k)^{(1-r^p)}\right) \quad \text{or} \quad x'_k = x'_k - \left((x_k - x_{\min k})^{(1-r^p)}\right)$$

with $p = (1 - \frac{t}{\mathrm{NGE}})^5$ and NGE is the maximum generation number and $x_{\max}$ and $x_{\min}$ are the maximum and minimum values for x_k and r is a random number between 0 and 1.

We proceeded from literature-based experiments to initialise for the genetic operators, and made experimental evaluations to find the best parameters. Performed trials helped to decide for the crossover probability at 60%. This rate applies to all of the three (discrete, continuous, and simplex) crossovers, where each is applied with 1/3 of the global crossover probability. ($P_c = P_d = P_s = (1/3) * P_C$). The mutation probability is supposed to be too drastically weak to make reliable the optimisation procedure. As a matter of fact, a 50% choice of the mutation probability is equivalent to making a pure random search. Following basic reference choices for mutation probability[18,23], we eventually adopted the probability 0.01.

The population size must be important to obtain a efficient selection in all promising zones of the search space and good distribution for the values of the objective function. Many tests were performed and we end up to a population size equals to $50 * \mathrm{ndf}$ (ndf: number of degrees of freedom).

For the maximum number of generations, it is necessary to have a sufficient number of generational loops (for instance, the mutation operator functions better only if it has a plenty of time!). A typical choice for the maximum number of generations is 500.

Figure 3 displays a flowchart diagram for the general architecture and operation of the Simplex-Genetic Algorithm.

5. SIMULATION RESULTS AND COMPARATIVE STUDY

In our simulation program, we realised a comparative study between the developed hybrid Genetic-Simplex Algorithm and the configuration space sweeping approach, on the model of a STAUBLI RX-90 robot. A schematic representation of this robot is given in Figure 1.

Geometric, internal parameters (masses, centers of mass, and inertia parameters), and kinematic parameter limits (maximum velocities and accelerations) are given in Tables I, II, and III. The first joint is rotational, and its axis is parallel to the gravity direction. This articulation does not occur in the dynamic model[25]. Hence, the maximum dynamic stress does not depend on q_1 and the search space is $3 * (n-1)$dimension (n is the number of degrees of freedom of the manipulator).

The reference payload is characterized by its mass and the two components of its position vector with respect to the last frame. To insert this payload in the last segment (Figure 4), we introduce the following notations: z, y, the second and third component of the payload position vector in the frame

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

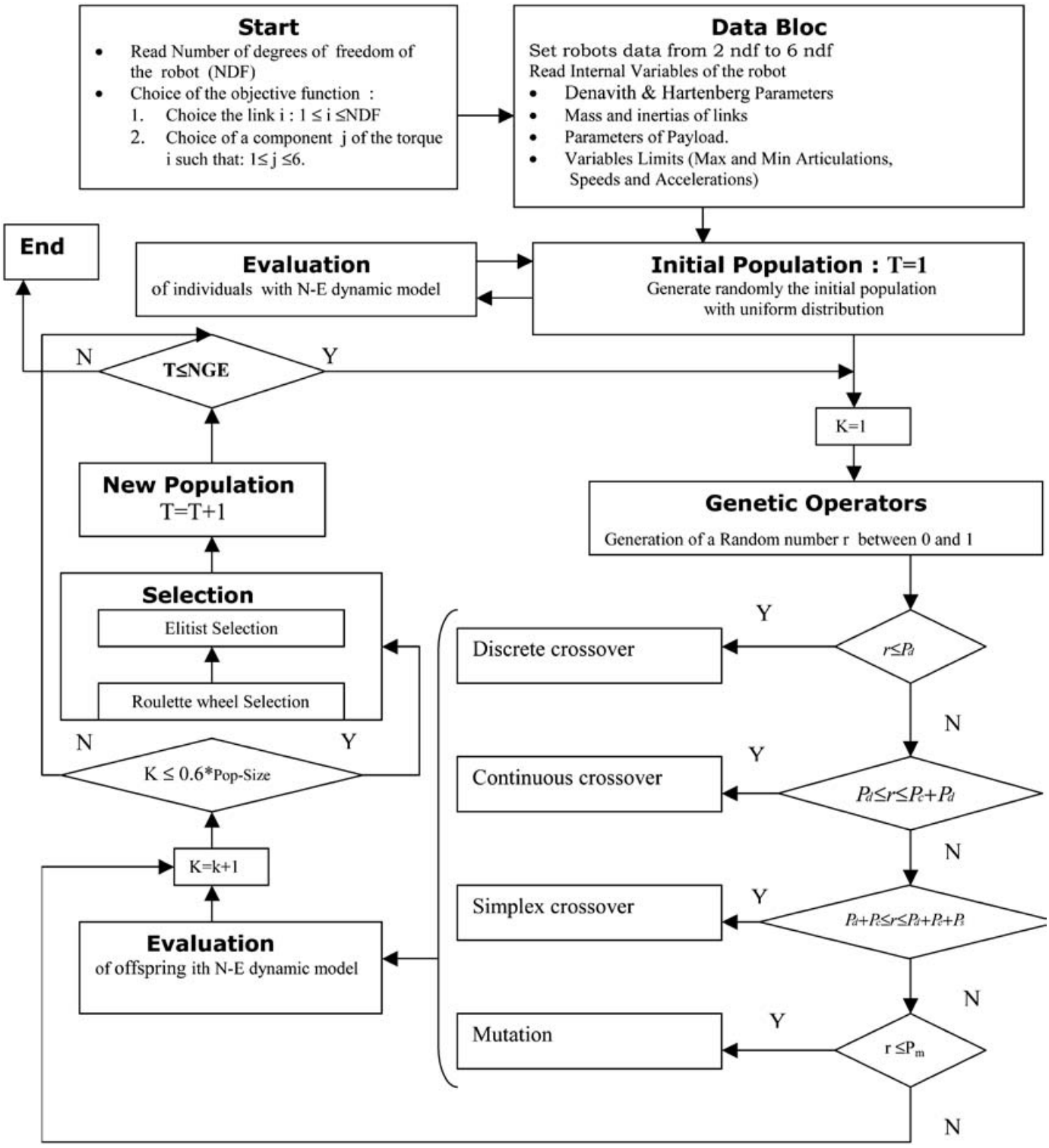


Fig. 3. Flowchart for the hybrid Simplex-Genetic Algorithm. **Nomenclature:** N-E Newton-Euler, NDF number of degrees of freedom, P_m Mutation Probability, Pd Discrete Crossever probability, Pc Continuous Crossover Probability, Ps Simplex Crossover Probability, r Random number, NGE Maximum Number of Generations, K Population Candidate Size, T generation increment.

R_6, the first component is null (x= 0)

m_p the payload mass,
M_n mass of the last segment
M'_n mass of the last segment plus payload
Lgc_n vector of the payload center of mass
Lg_n vector of the center of mass of the last segment
Lg'_n vector of the center of mass of last segment plus payload
Jg'_n inertia parameter of the last segment plus payload with respect to the center of mass.
Jo_n inertia parameter of the last segment with respect to its frame center.
Jo'_n inertia parameter of the last segment plus payload with respect to its frame center.

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

Table I. Kinematic Limits (articular position, speed, and acceleration limits).

Links	1	2	3	4	5
Link Type	R	R	R	R	R
$(q_j)^{(0)}_{\min}$	−140	−120	−90	−100	−90
$(q_j)^{(0)}_{\max}$	140	120	90	100	90
$(\dot{q}_j)_{\max}$	1	1	1	3	3
$(\ddot{q}_j)_{\max}$	1.25	1.25	1.25	3.75	3.75

Table II. Denavit-Hartinberg Parameters.

	Parameter			
Segment	α_i	a_i	r_i	θ_i
1	0	0	0	θ_1
2	90°	0	0	θ_2
3	0	0.15	0.432	θ_3
4	−90°	0.432	0.02	θ_4
5	90°	0	0	θ_5

The insertion procedure has been used for both approaches, the sweeping and SGA. It is performed as follows:
(i) Let's consider the matrix:

$$X_c^* = \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix}.$$

The inertia parameter of the payload with respect to the center of the frame is $J_{con} = -mc \cdot [X_c^*] \cdot [X_c^*]$

The inertia parameter of the last segment plus payload with respect to the center of the frame is

$$Jo'_n = Jo_n + Jc_n.$$

(ii) The coordinates of the center of mass of the set end-effector + payload:

$$\mathrm{Lg}'_n = (M_n \cdot \mathrm{lg}_n + m_c \cdot \mathrm{lgc}_n)/(M_n + m_c)$$

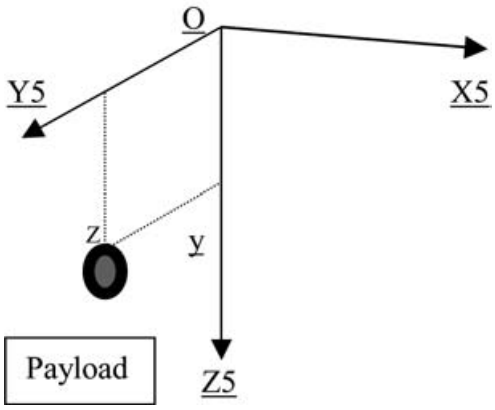


Fig. 4. Payload Layout.

(iii) The inertia matrix of the last segment plus payload with respect to the center of mass:

$$\mathrm{lg}'^*_n = \begin{bmatrix} 0 & \mathrm{lg}'_n(3) & -\mathrm{lg}'_n(2) \\ -\mathrm{lg}'_n(3) & 0 & \mathrm{lg}'_n(1) \\ \mathrm{lg}'_n(2) & -\mathrm{lg}'_n(1) & 0 \end{bmatrix}$$

$$Jg'_n = Jo'_n + M'_n \cdot \lfloor \mathrm{lg}'^*_n \rfloor \cdot \lfloor \mathrm{lg}'^*_n \rfloor.$$

The following numerical values, $m_p = 10\,\mathrm{Kg}$, $x = 0\,\mathrm{mm}$, $y = 50\,\mathrm{mm}$, $z = 100\,\mathrm{mm}$ had been used along with a uniform discretization of 10 knots, which gives $N_{\mathrm{conf}} = 2^5 \cdot 2^5 \cdot 10^4 = 1024 \cdot 10^4$ possible configurations to sweep. The simulated configurations using the hybrid Genetic Algorithm is $N_{\mathrm{confGP}} = 2 \cdot 10^5$.

According to Figure 5(a–f) the graphs show that the average adaptation of the population starts with weak values and tends toward the maximum, which is still weaker than the best chromosome. We also notice the fast evolution of the best chromosome to its maximum. The first augmentations are important for the components $F(1, 3)$, $N(1, 3)$, $F(5, 2)$, $N(1, 2)$, and $F(4, 1)$, but beyond the hundredth generation, the augmentation got weaker. The component $N(5, 2)$, however, reaches its maximum value only after the 450th generation. This suggests that we can achieve time saving with a better tuning of evolution parameters of the SGA,

Table III. Internal Parameters of the robot.

		Segments				
Parameters		1	2	3	4	5
Mass (Kg)		9.29	5.01	4.25	1.08	0.63
Centers Of Masses [m]	X	0	0	0	0	0
	Y	0.0018	−0.105	0	0.0092	0
	Z	−0.105	0	−0.6447	−0.0054	0.056
Inertias [Kgm^2]	Ixx	0.276	0.108	2.51	0.002	0.003
	Iyy	0.255	0.018	2.51	0.001	0.003
	Izz	0.071	0.100	0.006	0.001	0.0004
	Ixy	0	0	0	0	0
	Ixz	0	0	0	0	0
	Iyz	0	0	0	0	0

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

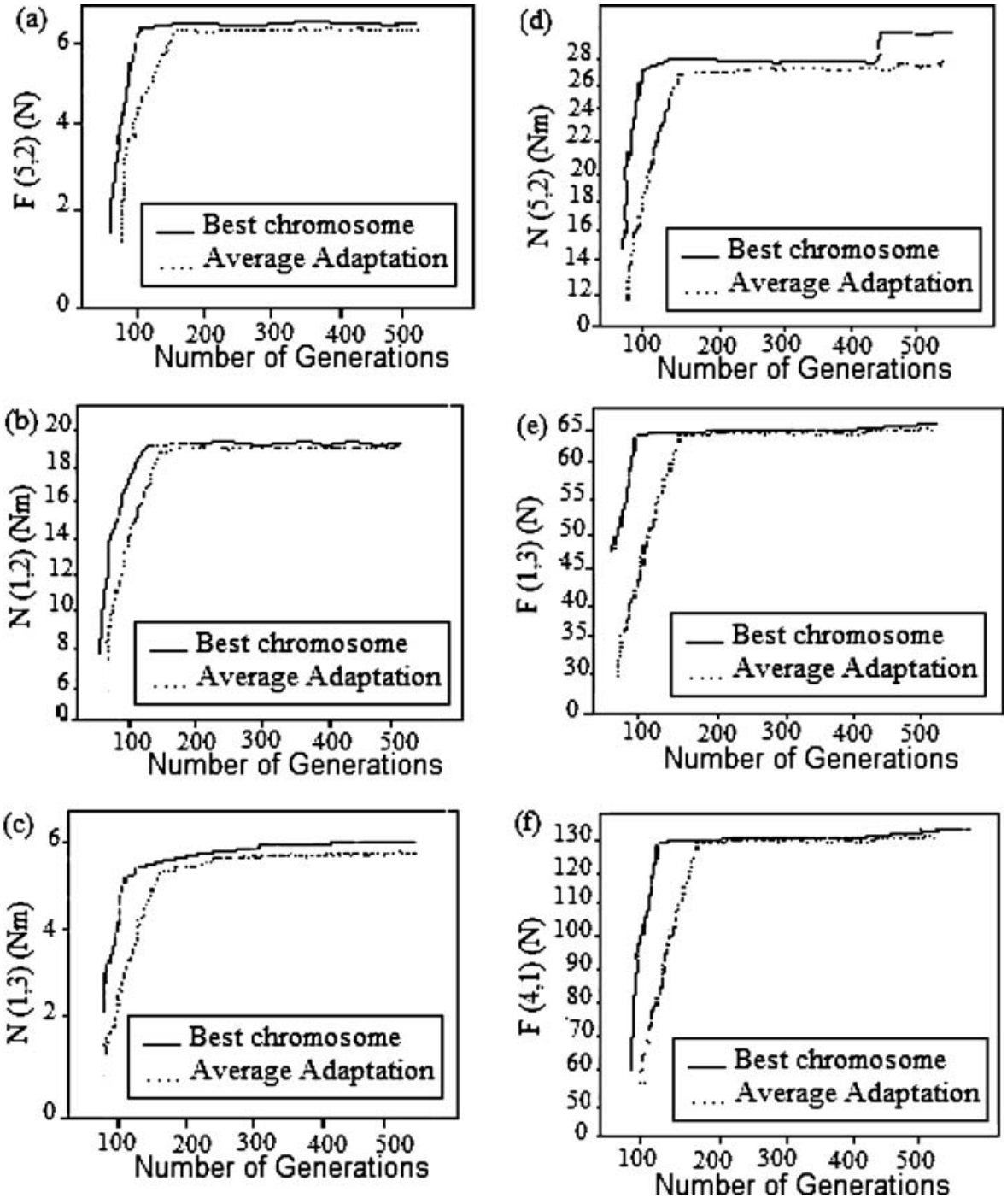


Fig. 5. Evolution of Best Chromosomes and Average Adaptation. Variation of (a) *F(5.2)*, (b) *N(1.2)*, (c) *N(1.3)*, (d) *N(5.2)*, (e) *F(1.3)*, (f) *F(4.1)*.

(especially the population-size, number of generations, and crossover and mutation probabilities).

Figure 6(a–F) displays the MDS (Maximum Dynamic Stress), position, velocity, and acceleration variations for inverse sweeping and simplex-genetic algorithms for the fifth link as functions of six wrench components (composed of forces and net torques). One notices that each wrench component has a corresponding maximum value within specific and proper configurations. Figures 6, part related to the Simplex-Genetic approach, are given with maximum velocity and accelerations. It is known that the maximum dynamic stress appear when the robot works at its possible maximum acceleration, and not necessarily with its 'theoretical' maximum accelerations. This implies that the simplex-genetic approach does not ensure the global optimum. Nonetheless, it gives good solutions within lower processing time.

The whole comparison between the two simulated approaches shows the superiority of the SGA simile to the inverse sweeping. The ratio of maximum dynamic stress search with a hybrid simplex-genetic algorithm, as compared to that with sweeping algorithm, varies from 0.97 to 1.07. First and foremost, the configurations of maximum dynamic stresses are different from one method to another.

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

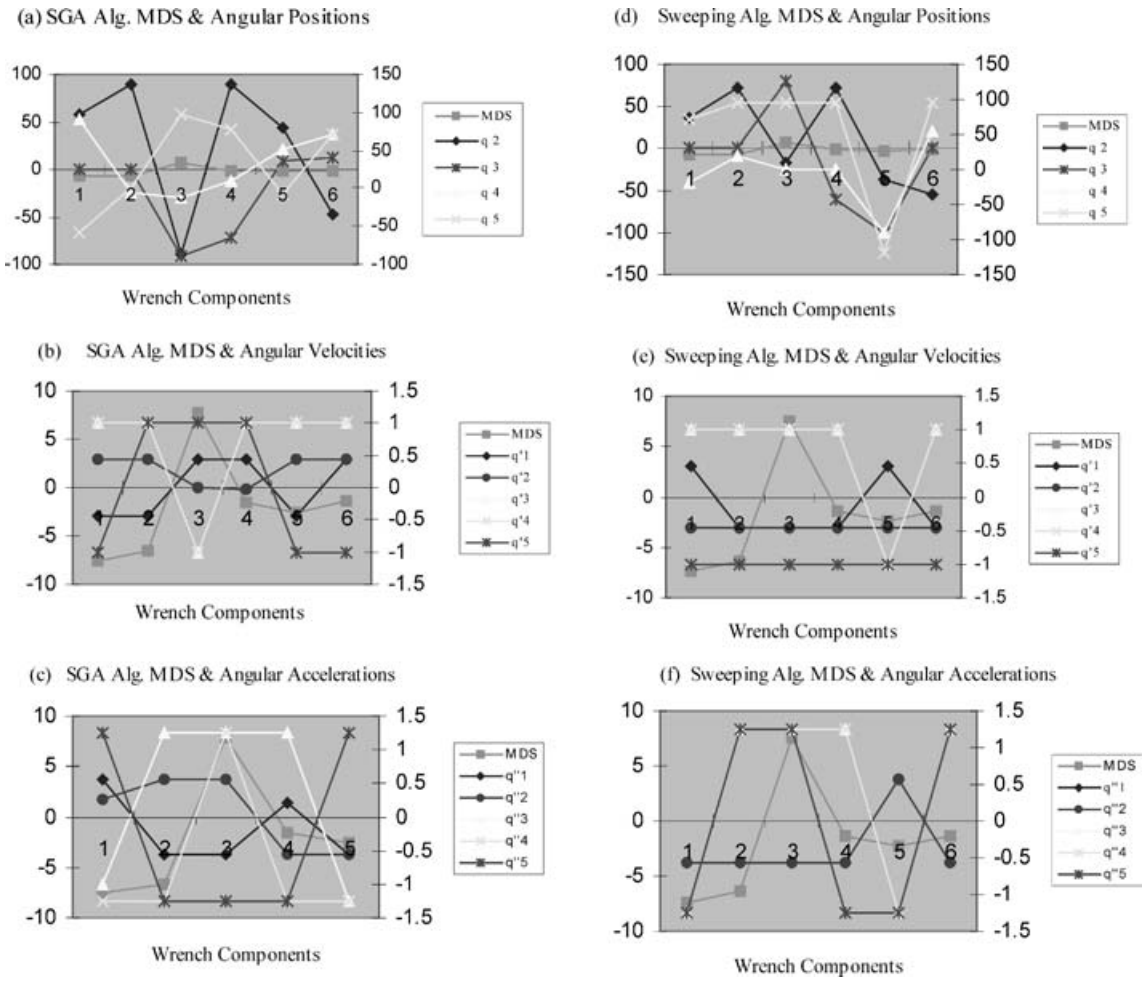


Fig. 6. Variations of Maximum Dynamic Stress (MDS), Sweeping vs. SG algorithms. Variation of (a) *F(5.2)*, (b) *N(1.2)*, (c) *N(1.3)*, (d) *N(5.2)*, (e) *F(1.3)*, (f) *F(4.1)*.

Overall, we noticed that with inverse sweeping, over the thirty components of five links, we have five computed components of maximum dynamic stress, that are quite superior to those found with a hybrid genetic algorithm. Here SGA can be improved by a judicious choice of evolution parameters associated to each component of the dynamic stress.

Nevertheless, in mechanical design, the safety coefficient ensuring the reliability of the designed mechanical system is always of order 1.5. Then obtained results with SGA are still acceptable.

Second, the evaluation of one or more maximum dynamic stresses with the sweeping approach requires the simulation of $10.24 * 10^6$ possible configurations, whereas with SGA the number of required simulated configurations for a link parameter component is of $2 * 10^5$. The smallness of the number of simulated configurations with a SGA simile to sweeping approach is given by the proportions:

- For one component: $\varepsilon = \frac{2 * 10^5}{10 * 24 * 10^6} = 0.0195$
- For the 30 components: $\varepsilon = 30 * 0.0195 = 0585.$

In terms of CPU processing time, the complete running of the SGA requires 165 sec. to evaluate one component of one torque and the global CPU time for all maximum dynamic stresses necessitates 4950 sec. The sweeping algorithm evaluates the components of all torques simultaneously, and requires 7437 secs. The gain in CPU time with a SGA simile to inverse sweeping algorithm is over 33.5%.

6. CONCLUSIONS

We have presented a hybrid simplex-genetic algorithm with its application to the problem of maximum dynamic stress induced by a reference payload for a manipulator arm. The developed simplex-genetic algorithm was compared to the conventional sweeping approach. We have presented the problem, the modelling aspects of the maximum dynamic stress and the inherent complexity of the optimisation problem associated with it. SGA operates by two crossover operator modes: standard and simplex with real coding of chromosomes and applies an enlarged sampling space with a two phases selection: The first applies an elitist selection, then the new population is determined by a roulette wheel Selection.

SGA has given better results for dynamic stress evaluation as compared to inverse sweeping of the configurations space.

Short-term outlooks of the presented work concern, firstly, of introducing a better tuning of evolution parameters of

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

SGA. Secondly, an increasing search quality by introducing neural networks based learning schemes. Thirdly, hybridising SGA with several proved deterministic optimisation algorithms to achieve a better exploitation of the chromosomes population.

References

1. D. Tesar, "Where is the field of robotics going?", *ICRAM' 99*, Istanbul, Turkey (May 26–28, 1999) pp. 24–26.
2. J. J. Craig, *Introduction to Robotics, Mechanics and Control*' 2nd ed. (Reading, MQ,: Addison-Wesley, 1989).
3. P. K. Allen, A. Tomcenko, B. Yoshimi and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand – Eye system", *IEEE Trans. Robot. Automat.* **9**, 152–165 (Apr., 1993).
4. R. O. Ambrose and M. A. Diftler, "The minimum form of strengths in serial, parallel and bifurcated manipulators", *Proc. of the IEEE Int'l. Conf. on Rob. and Aut.*, Leuven Belgium (May, 1998) pp. 1334–1339.
5. O. Chocron "Conception évoltionnaire des systèmes robotiques", *Doctorate Dissertation* (Université Paris VI, 2000).
6. S. K. Agrawal and T. Veeraklaew, "Designing robots for optimal performance during repetitive motion", *IEEE Trans. On Robotics and Automation* **14** (5), 771–777 (April, 1998).
7. K. L. Doty, C. Melchiorri and C. Bonivento "A theory of generalized inverses applied to robotics", *Int. J. Robotic Research* **12** (1), 64–79 (Feb., 1993).
8. Y. Farhang Sakhitab, "Contribution à la simulation du comportement dynamique des mécanismes tridimensionnelles, application à la commande des robots manipulateurś," *Thèse Docteur-Ingénieur* (ENSM Paris, 1983).
9. T. Yoshikawa, *Foundations of Robotics: analysis and control* (MIT Press, Cambridge, 1990).
10. A. Khoukhi, "An optimal time-energy control design for a prototype educational robot", *Robotica* **20**, Part 6, 661–671 (2002).
11. B. Siciliano and L Sciavicco, *Modelling and Control of Robot Manipulators* (McGraw-Hill, 1995).
12. J. Angeles, *Fundamentals of Robotic Mechanical Systems, Theory, Methods and Algorithms* (Springer-Verlag, New York 1997).
13. F. L. Chernousko, N. N. Bolotnik and V. G. Gradetsky, *Manipulation Robots: dynamics, control and optimization* (CRC Press, 1994).
14. L. Dixon, J. Gomulk and Herson, *Optimization in Action* (Academic Press, 1976).
15. R. Horst, P. M. Pardalos and N. V. Thaoui, *Introduction to Global Optimisation* (Kluwer Academic Publishers, London, UK., 1995)
16. G. James, *Advanced Modern Engineering Mathematics* (Addison-Wesley, New-York, 1994).
17. A. Moualek and A. Khoukhi, "A new class of global optimisation and search algorithms" (Submitted to Cybernetics and Systems An International Journal, 2004).
18. K. F. Man, K. S Tang, S. Kwong and W. A. Halang, *Genetic Algorithms for Control and Signal Processing* (Springer-Verlag, London, 1997).
19. A. Khoukhi and A. Moualek: "Genetic agents, a new class of quasi-biologic algorithms", *Int. J. Hybrid Systems* **2**(4), 383–399 (Dec., 2002).
20. D. E. Goldberg, R. Harik and G. Lobo "The compact genetic algorithms", *IEEE Trans. On Evolutionary Computation* **3** (4), 287–297 (Nov., 1999).
21. L. J. Fogel, *Intelligence through Simulated Evolution - Forty years of Evolutionary Programming* (John Wiley & Sons, New York, 1999).
22. J. Gottlieb, *Evolutionary Algorithms for Constrained Optimisation Problems* (Shaker-Verlag, Aachen, 2000).
23. M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design* (John Wiley & Sons, New York, 1997).
24. J. E. Smith and T. C. Forgarty, "Operator and parameter adaptation in genetic algorithms", *Soft Computing* **2**, 81–87, (1997).
25. W. Khalil and E. Dombre, *Modélisation et commande des robots*, 2eme edition (Hermes Ed., Paris, 1999).

APPENDIX A. CROSSOVER OPERATOR

For each reproduction group, we generate a random number r between 0 and 1, and let Pc, Pd, and Ps the continuous, discrete, and simplex probabilities (respectively).

If $r \leq Pd$, then apply discrete crossover by randomly choosing for each gene of the offspring the corresponding value from one of $n+1$ parents (Fig. 7)

If $Pd < r \leq Pc + Pd$, then apply continuous crossover operator, by taking for each gene of the offspring the average of the corresponding $n+1$ parent genes (Fig. 8)

If $Pd + Pc < r \leq Pd + Pc + Ps$, then apply the simplex crossover according to the following schema (Fig. 9): First, to properly apply the simplex crossover, we transform the $(n+1)$ chosen individuals into dimensionless elements. We note $\overline{X_j}$ to represent the dimensionless element X_j

$$\overline{X_j} = \frac{X_j}{X_{\max} - X_{\min}} \quad j = 1, \ldots, n+1$$

$\overline{X_j}$	The vector of dimensionless parameters.
X_j	The vector of dimensioned parameters
$X_{\max}$	Maximum bound value of the chromosome's parameters
$X_{\min}$	Minimum bound value of the chromosome's parameters

Each element of the descendent vector is computed through two steps:

The first step is to select a dimensionless individual from the bloc of $(n+1)$ simplex crossover individuals. This selection is randomly generated according to a random trial of an integer $k_1 \in [1, n+1]$

The second step determines one element among the elements of the selected individual from the first step. The selection is randomly generated according to a second random integer

$$k_2 \in [1, 3 \times (n-1)]$$

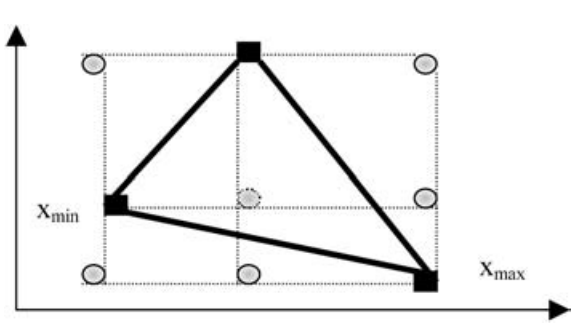


Fig. 7. Discrete crossover on a 2 dimensional space. Black squares: Parents, Surrounded bubble points: Offspring.

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press

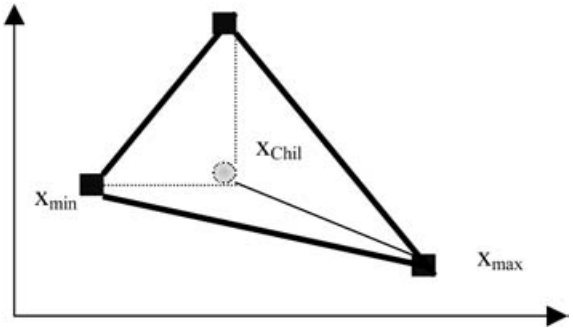


Fig. 8. Continuous crossover on a 2 dimensional space. Black squares: Parents, Surrounded bubble point: Offspring.

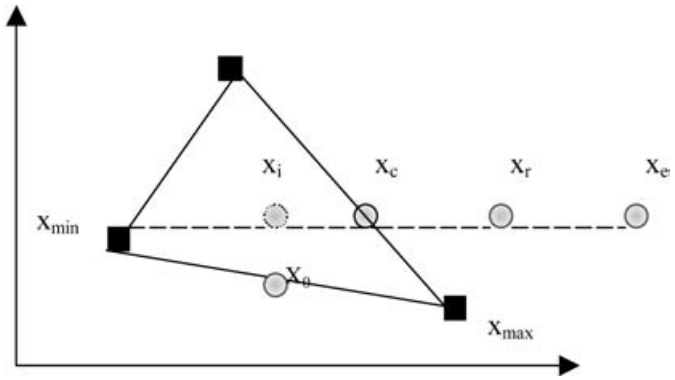


Fig. 9. Simplex crossover on a 2 dimensional space. Black squares: Parents, Surrounded bubbles: Offspring.

Once, this dimensionless element is calculated we transform it back to a dimensioned one.

The offspring will replace the worse individual in the reproduction group.

The procedure is illustrated through the following pseudo-code:

1)- Find the best $X_{\max}$ and the worst $X_{\min}$ individual among all individuals in the considered group

2)- Compute the center X_c of all individuals of the group except the worst:

$$X_c = \frac{\sum_{i\neq \min}^{n+1} X_i}{n}$$

3)- Compute the reflected point from the worse individual:

$$X_r = X_c + (X_c - X_{\min})$$

If X_r is better than $X_{\max}$(i-e it makes higher the fitness function value), **Then**

4)- Create a new point called a "extended point" (X_e) such that: $X_e = X_r + (X_r - X_c)$

if X_e is better than X_r, **Then**

Pick X_e as offspring.

Else Pick X_r

Else if X_r is not better than $X_{\max}$, but nevertheless, it is better than $X_{\min}$, **Then**

The offspring is X_r

Else If X_r is not as fitting as $X_{\min}$, **Then**

5)- Calculate the "contracted point" X_i defined as:

$$X_i = \frac{X_{\min} + X_c}{2}$$

If X_i is better than $X_{\min}$, **Then**

The offspring is X_i

Else the offspring will be a contracted point in the best direction: $X_0 = \frac{X_{\min}+X_{\max}}{2}$. The resulting offspring will replace the worse individual of the group.

https://doi.org/10.1017/S0263574704000281 Published online by Cambridge University Press