Robotica (2008) volume 26, pp. 323–330. © 2007 Cambridge University Press
doi:10.1017/S0263574707003931 Printed in the United Kingdom

Evolutionary design of modular robotic arms

O. Chocron†,*

†*Laboratorie de recherche en mécatronique, Ecole nationale d'ingénieurs de Brest, Technopôle Brest-Iroise, CS 73822 Brest Cedex 3, France*

(Received in Final Form: September 5, 2007. First published online: October 10, 2007)

SUMMARY

This paper proposes a method for task based design of modular serial robotic arms using evolutionary algorithms (EA). We introduce a 3D kinematics and a global optimization for both topology and configuration from task specifications. The search features revolute as well as prismatic joints and any number of DOF to build up a solution without using any design knowledge. A study of the evolution dynamics gives some keys to set evolution parameters that enable artificial evolution. An adapted algorithm dealing with the topology/configuration search tradeoff is proposed, descibed, and discussed. Illustrations of the algorithms results are given and conclusions are drawn from their analysis. Perspectives of this work are given, extending its reach to control and complex system design.

KEYWORDS: Modular robots; Task-based design; Kinematic synthesis; Evolutionary optimization.

1. Introduction

Because the trend in robotics is that missions are growing in diversity, complexity and constraints, new solutions are necessary to answer the issue of adaptation. Modular Robotic Systems (MRS) have been proposed for adapting robot morphologies to given tasks.[1–3] MRS are minimal systems, able by recombining their modules, to adopt a task qualified morphology. The paradigm used here is that task diversity is answered by modular configuration diversity.[4] Some important contributions have been made to describe these systems in computer programs using a design process,[5] an incidence matrix,[6] or a generative representation allowing open-ended design.[7]

The task-based design problem raised is to find the right assembly of modules that fits the desired task. Both deterministic and stochastic methods have been investigated to address this combinatorial optimization problem known as NP complete.[8] Combinatorial explosions and irregularities of functions representing the assembly (topology) and task specifications (configuration) make deterministic methods uneasy if not unworkable.[9] The main failing of a deterministic method is that it exploits heuristics that think locally (module features) but act globally (whole robot). As a consequence, for a given problem the rule-based optimization always yields to the same set of solutions,[10] making it very problem dependent.

* Corresponding author. E-mail: chocron@enib.fr

Methodologies based on stochastic optimization search for best solutions with at least partially random operators. When considering multiobjective optimization; progressive, partial, or global methods are proposed. Progressive methods try to match sequentially different constraints, to reach partially optimized solutions.[11,12] Unfortunately, strongly coupled constraints involve some backtracking, thus severely slowing down the search process. Partial methods consist in applying optimization only on some parts of the problem as the inverse kinematics,[13] or the degrees of freedom (DOF) minimization,[14] leaving other design parameters to the user. The problem with this second approach is that there is no guarantee that the user-computer interaction is beneficial. Global methods consist in trying to match simultaneously all constraints while optimizing some criteria.[15] This last approach is the most difficult but represents a more promising one because of the exploration capabilities brought by the association of random operators and parallel optimization. Latest works based on this approach yield to very interesting results since it has been applied to physical robots and thus, proved that the gap between simulation and reality can be stepped across.[16]

Our approach which is stochastic, global and with no fundamental restrictions has been explored with promising results for evolutionary solving of robotic problems as the inverse kinematics problems (IKP) or synthesis of general manipulators.[17–20]

We propose a complete frame of task based synthesis and three evolutionary algorithms to address the problem of kinematic design for serial modular manipulators from task specification and under nonexplicit constraints (obstacles). In the following sections, we first layout the problem statement and propose an adapted modular kinematics as well as evaluation criteria. Three implementations of artificial evolution for the synthesis problem and their results are proposed that give an insight about involvement of field knowledge in the success of evolutionary design.

2. Synthesis Problem

The synthesis problem consists in finding the fittest modular robot constructed from an assembly of links and joints (topology) as well as sets of joint parameters (configurations) according to given manipulation tasks. In our work, we considered serial topologies and joint configurations (poses) which cover most of industrial needs. Revolute or prismatic joints and under-actuated or redundant manipulators are allowed.

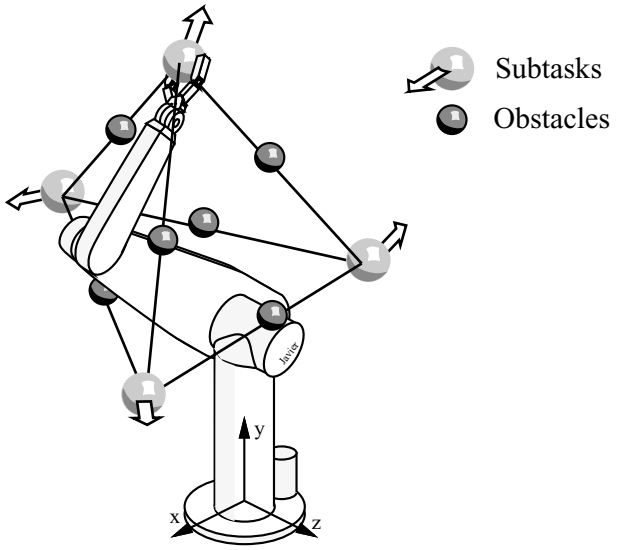

Fig. 1. Task specification and a PUMA solution.

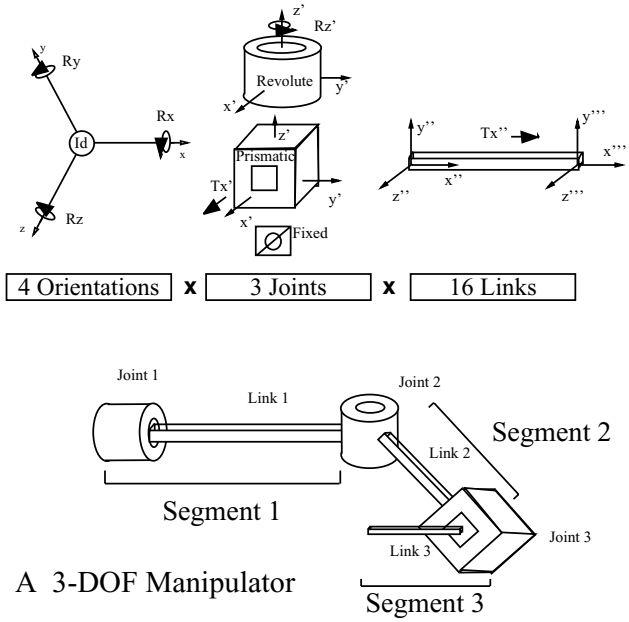

Fig. 2. Module assembly.

2.1. *Task specification*

The global task is specified as a set of n subtasks which are 3D end-effector poses including position and orientation. For our synthesis implementations, the global task area is a tetrahedron (Fig. 1).

The workspace is cluttered with n_o obstacles (represented by spheres) to be avoided by the robot segments. The topology has undergone evolution to perform the global task that is evaluated through a multi-objective function relying on four objectives:

- Reaching each subtasks with end-effector
- Avoiding collisions with obstacles
- Avoiding kinematic singularities
- Minimizing the manipulator

2.2. *Modular kinematics*

The manipulator is constituted by two classes of modules: structural (links) and articular (joints). The mobility of the mechanism is given by the total number of DOF. Two 1-DOF joint types (R for revolute and P for prismatic) and one 0-DOF joint (F for fixed) are available, as well as a set of links of different lengths. The links geometry is chosen linear since more complex geometries (as elbows) can be obtained by combination of links and F-joints. Table I shows the available modules with their given features.

As we consider only serial manipulators, assembly is made with successive segments (S). Each segment is constituted by a joint (R, P, or F) and by a link (L). Before connecting a new segment, it is possible to rotate it by a quarter turn (+90 degrees) about any basis axis (x, y, or z). The segment and manipulator assembly is illustrated in Fig. 2 and the design parameters are described in Table II.

Table I. Robotic modules.

Modules	Class	DOF	Features
Link	structural	NA	set of lengths L_i
R-joint	articular	1	2π rotation range
P-joint	articular	1	L_i translation range
F-joint	articular	0	–

Table II. Design parameters for a segment.

Parameter	Domain	Symbol	Values
Orientation	topology	α	$R_x\ R_y\ R_z\ Id$
Joint type	topology	λ	R, P or F
Joint value	configuration	θ	variable
Link length	topology	l	variable

The basic transformation from a link to the following is given by the 4×4 homogeneous matrix:

$$\boldsymbol{T}_i^{i+1} = \left(R_{x,y,z}^{90} \text{ or } I_d\right) \times \left(R_z^{\theta} \text{ or } P_x^{\theta} \text{ or } I_d\right) \times P_x^{l} \quad (1)$$

R_a^b: 4×4 matrix for rotation about axis a with angle b
P_a^b: 4×4 matrix for translation along axis a with offset b
I_d: 4×4 identity matrix

The structure equation of the whole mechanism may be computed as follows:

$$\boldsymbol{T}_0^e = \prod_{i=1}^{i=n} \boldsymbol{T}_i^{i+1} \quad (2)$$

where the matrix $\boldsymbol{T}_0^e$ represents the end-effector pose.

The concatenation of segment parameters are the design parameters for the synthesis problem. That specifies how the robotic modules can be combined to constitute segments and thus, the robot. This kinematics enables a wide range of diverse topologies (Fig. 3), thus matching any serial manipulator.

https://doi.org/10.1017/S0263574707003931 Published online by Cambridge University Press

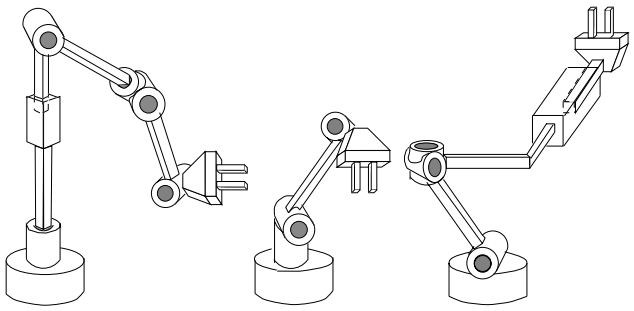

Fig. 3. Various serial modular manipulators.

2.3. Evaluation

Evaluation is the key point of optimization since it drives the whole search process through candidates toward solutions and thus, deserves a particular attention. Evaluation in optimization uses an objective function F to be maximized (or minimized). F has to be computable and relevant for any candidate the algorithm is able to generate. The evaluation computing time is a critical point in optimization since it is performed a large number of times.

There are three kinds of functional criteria that need to be answered specifically:

- objective : has to be achieved eventually;
- constraint : has to be respected at all cost;
- preference : has to be sought as best as possible.

The objectives give rise to main objective function F and are usually selective all over the search. The constraints can be handled two ways. The first consists in eradicating any solution that violates them. Is it rather brutal but represents a good way when we know that studying such a solution is just a waste of time. The second way is to penalize (decreasing F) the violating candidate according to the degree of violation. It is useful when there is reasons to think the candidate will yield a good solution further in the search. Hard constraints are preferably handled with eradication and soft ones with penalization. Eradication activity is high at the beginning of the optimization (massively rejecting ill candidates), then decreases to near zero very fast. Penalization activity is moderate during the whole optimization depending on the level of severity of the penalty function. To finish, preferences are considered as secondary objectives and should be sought when the main objective is achieved. The problem is that it is not always clear whether or not the preferences are intimately coupled with the objectives. Actually, it is possible that it could be drastically improved with a very small loss on the primary objectives or of time. From task specifications, we can define:

- Objectives
  - Linear distance from subtask (L)
  - Angular distance from subtask (A)
- Constraints
  - Reachability (R)
  - Obstacle Proximity (O)
  - Dexterity Measure (D)
- Preferences
  - Minimizing mass and actuators (I)

2.3.1. Linear and angular distances. Linear and angular distances are used to assess the end-effector configuration, according to the considered subtask. The Euclidean norm of position vector is used for linear distance and the L_1 norm of a difference matrix is computed for angular distance. L for linear and A for angular distances are dimensionless criteria defined as:

$$L = \frac{\|\boldsymbol{X_d} - \boldsymbol{X_e}\|}{\|\boldsymbol{X_d}\|} \geq 0$$

$$A = \frac{\|R_d - R_e\|_1}{\|R_d\|_1} \geq 0$$

$\boldsymbol{X}_d$: subtask position vector
$\boldsymbol{X}_e$: end effector position vector
R_d: desired orientation matrix
R_e: effector orientation matrix
$\|M\|_1 = \sum_{ij} |M_{ij}|$: L_1 matrix norm of M

2.3.2. Reachability. Reachability is defined as the max value of missing reach to get any subtasks. If it is positive, then the objective function (F) is set to zero. In this way, the optimization can go on with other manipulator topologies. This means that R is checked first before any other criteria since a failure here means eradication.

$$R = \max_{i=1..n_t} \left(d_j - \sum_{i=1}^{n} l_i \right) \tag{3}$$

l_i: length of ith link (doubled if prismatic)
d_j: distance of jth task position from base
n: maximum number of DOF
n_t: number of goal frames (subtasks)

2.3.3. Obstacle proximity. An approximation had to be designed so the computation time is not prohibitive. We check whether each link is interacting or not with each obstacle by using a security ellipsoid defined by the link and a user defined security distance (Fig. 4). If the obstacle is inside the security ellipsoid, there is interaction and the penalty function O is increased by an amount proportional to the

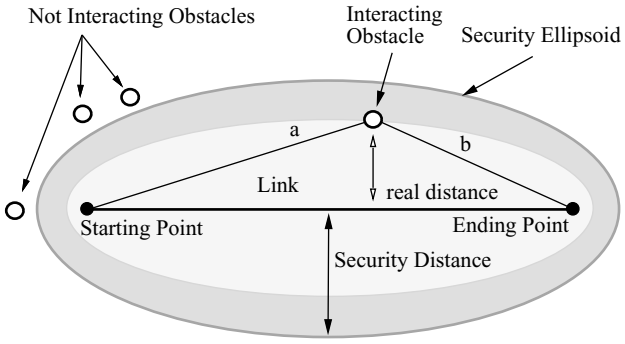

Fig. 4. Obstacle avoidance and security ellipsoid.

https://doi.org/10.1017/S0263574707003931 Published online by Cambridge University Press

penetrating distance.

$$\text{if Obstacle} \notin SE_i \Rightarrow cp_{ij} = 0$$

$$\text{if Obstacle} \in SE_i \Rightarrow cp_{ij} = \frac{(s_d - d_{ij})}{s_d}$$

$$O = \sum_{i=1}^{n} \sum_{j=1}^{n_o} cp_{ij} \geq 0$$

SE_i: security ellipsoid for ith link
d_{ij}: distance of ith link from jth obstacle
cp_{ij}: collision penalty for ith link and jth obstacle
s_d: security distance
n_o: number of obstacles

2.3.4. Dexterity. The dexterity D is evaluated with the Yoshikawa manipulability index.[21] This index represents a distance of the joint configuration from a kinematic singularity. Singularities are avoided with minimizing D defined by:

$$w = \sqrt{Det(J J^t)}$$

$$D = \frac{1}{1 + w} \geq 0$$

w: manipulability index
J: manipulator Jacobian matrix

2.3.5. Involved modules. The involved module criterion I is based on total link lengths as well as the number and type of joints. Each active joint is weighted according its mass (based on link length for prismatic joints). I is defined as

$$I = \frac{\sum_{i=1}^{n} l_i^* + m_a * \lambda_i}{d_j} \geq 0 \qquad (4)$$

λ_i: set to 0, 1 or 2 if ith joint is F, R, or P
l_i^*: length l_i of ith link (or $2l_i$ for P-joints)
m_a: unit mass of revolute joint actuators
d_j: end effector distance from base frame

2.3.6. Objective function. All evaluated criteria are minimized since they are positive penalty functions. A positive (and bounded by 1) fitness function F to be maximized is designed from these criteria and used to evaluate candidate solutions. Each criteria is associated to a weight k_i which is user defined and allows to adjust priorities among the criteria. If R is negative, F is defined as

$$F = e^{-(k1*L+k2*A+k3*I+k4*O+k5*D)} \qquad (5)$$

k_i: Weight of ith criterion (let free)

The optimization goal is then to find a set of design parameters (DP) which maximizes F according to the priorities (k_i).

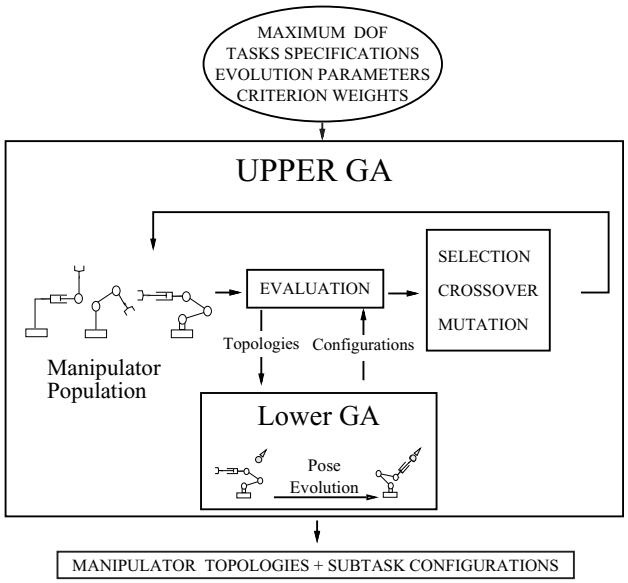

Fig. 5. A two-level genetic algorithm (TGA).

3. Adapting an Evolutionary Algorithm

The first idea coming using genetic algorithms (GA) is to use them as what they were made for: parameter optimization. Since we dispose of two kinds of design parameters (configuration and topology) and we need to solve the first ones to evaluate the others, we have designed a two-level genetic algorithm (TGA) to solve the two problems (Fig. 5).

The upper genetic algorithm evolves manipulator topologies and calls a lower level GA (searching subtask configurations) for evaluation. The basic problem in gathering all the genotype (structure and configurations) on a single binary chromosome is the inefficiency of genetic operators for long strings (more than 300 bits for our application). Longer the chromosome is, less effective is the crossover for instance, because it annihilates a large part of the genotype ordering for changing just one small part. When the algorithm tries to improve a task configuration for instance, others are likely to be discarded by a global action whatever their evolution achievement.

To avoid this problem, we propose to dispatch the genotype over several chromosomes. Each chromosome gathers some highly linked informations which so, are not disturbed by a global crossover. Figure 6 shows how genotype, evaluation and genetic operators are distributed over the robot. The topology and base position is considered as having a global range over the robot and is evaluated using the global fitness. Each configuration has its own chromosome and is not concerned with what happens to others.

Each chromosome undergoes its own local evaluation on which are based its genetic operators. In some ways, each chromosome can evolve with its own dynamics since it possesses its own operators. Global evaluation is based on all local evaluations and is used to select the overall candidate solution. This global selection is maintained because we want the algorithm to converge toward a coherent solution. The Multi-Chromosome Algorithm (MEA) includes only one evolution loop ans thus, is much faster than the TGA.

https://doi.org/10.1017/S0263574707003931 Published online by Cambridge University Press

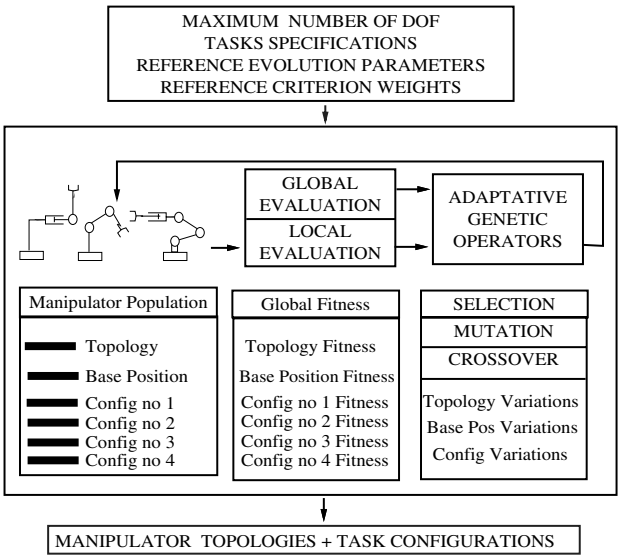

Fig. 6. Multiple-chromosome evolutionary algorithm (MEA).

3.1. Evaluation

Evaluation must be adapted to the new different chromosomes. We define thus three objective functions corresponding to three evaluations:

$$F_{topology} = e^{-k_R * R - k_I * I} (\text{or } 0 \text{ if } R > 0) \tag{6}$$

$$F^j_{cfg} = e^{-k_L * L^j - k_A * A^j - k_O * O^j - k_D * D^j} \tag{7}$$

$$F_{global} = F_{topology} * \tilde{F}_{cfg} \tag{8}$$

$\tilde{F}_{cfg}$: mean fitness of subtasks configurations

The evaluation of topology includes the global criteria R and I but includes the base position evaluation since R depends on it. The configuration evaluation concerns only the joint-values dependent criteria (L, A, O, and D) for each subtask and a mean evaluation is computed. These evaluations are used to adapt the action of the genetic operators to their chromosomes. The global fitness is used to select the robot (topoly and configuration) during evolution.

3.2. Encoding

Binary coding is used for topologies and real number coding for base position and configurations (Fig. 7). These choices have been made because non-valued parameters (as joint type) can be easily described by discrete values but float numbers allows to encode the configuration parameters without the loss of precision involved in discretization.

The topology chromosome does not change and keeps its operators and fitness function (based on R and I). The base position has been added to the genotype, to optimize the placement of the robot with regard to the subtasks. The base and joint configurations are encoded with real valued genes and put in separate chromosomes (Fig. 7). Using 6-digit floats numbers, we have $256^n \times 10^{6n*nt+18}$ candidate solutions for the global genotype.

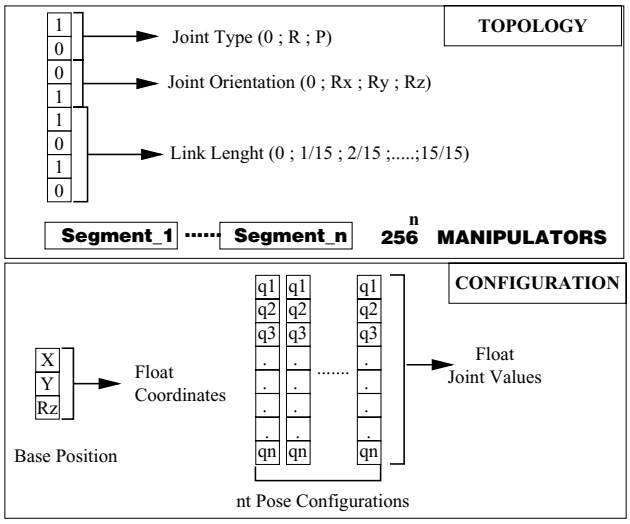

Fig. 7. Multiple-chromosome genotype.

3.3. Adaptive genetic operators

Genetic operators are successively applied to all chromosomes for each generation in a process similar to canonical GA.[22] The difference lies in the fact that the operators are adapted to the piece of genotype they manipulate. Each chromosome can be evaluated separately and then be modified according to its specific fitness. Since operators are specific to each chromosome, it becomes possible to adapt them to the evolution of their chromosome population. This dynamic adaptivity aims at giving the genetic operators an awareness about the local effect of their actions. In some ways, it is similar to replace optimization heuristics with closed loop control since the goal will not be to apply rules, but to control some criteria to maintain evolvability.

Selection pressure is relevant to the well known exploitation versus exploration dilemma of optimization. If a quick convergence, toward a single solution is needed we insist on exploitation, but if an extensive search, with numerous different solutions is necessary we advantage exploration. A good evolvability will be maintained if the selection pressure assures dynamically a good balance between exploitation and exploration. Increasing or decreasing the selection pressure is made by smoothing or emphasizing the fitness discrepancies. We designed an adaptive selection pressure that acts upon the fitness function with a hardness coefficient λ, controlled by the fitness standard deviation σ_f (Fig. 8).

If the solutions are widely scattered on the fitness landscape (large σ_f), we need to select more thoroughly and so, a hard selection pressure is required (large λ). In the other way, if most solutions are tightly clustered near a mean value, we

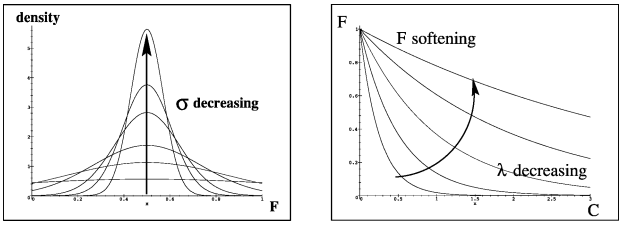

Fig. 8. Selection adaptation to diversity.

https://doi.org/10.1017/S0263574707003931 Published online by Cambridge University Press

let other individuals a better chance to survive and so, the fitness function is softened by decreasing λ. Thus, this adaptive scaling will help the selection operator to keep a diversified population which means a good evolvability. Meanwhile, this strategy has to be modified to force convergence toward high fitnesses to spend last resources on known solutions for ultimate improvements. So, we use also the mean fitness to harden the fitness function for high mean fitness.

$$\lambda = \lambda_o(\tilde{f} + \sigma_f) \;\; and \;\; fm = f^{\lambda} \tag{9}$$

λ_o: Reference selection pressure
$\tilde{f}$: Population mean fitness
fm: Modified individual fitness

To avoid the disruptive effects on crossover, we separated genotypes in several chromosomes, but we can do more. A uniform crossover is selected instead of the simple one point crossover for binary strings to allow exchange of middle pieces of genetic code. This allows to exchange only parts of the manipulator as a joint or a link length without breaking the rest of the code. For real encoding, the uniform crossover consists in exchanging genes (floats numbers) between both parents chromosomes. To prevent the configuration crossover to be too disruptive, we modify the permutation probability (initially 0.5) according to the local fitness of each configurations to reduce the highly fit parameters migration. Each manipulator is

$$f_\alpha^i = \frac{F_\alpha^i}{\tilde{F}_\alpha} \tag{10}$$

$$p_{\alpha\beta}^i = \frac{0.5}{f_\alpha^i * f_\beta^i} \tag{11}$$

F_α^i: Fitness of configuration i for robot α
$\tilde{F}_\alpha$: Fitness of configurations for robot α
f_α^i: Relative fitness of configuration i for robot α
$p_{\alpha\beta}^i$: Permutation probability for ith configuration

Adaptation of binary mutation is made by adjusting the mutation probability to according the topology fitness.

For float value chromosomes, a normally distributed random variable z of zero mean value is added to the float number according to evolution strategy principles.[23] This mutation is controlled by modifying the standard deviation σ of z according to the local fitness f_i.

- Binary Mutation

$$p_m = \frac{p_{m0}}{F} \tag{12}$$

- Float Mutation

$$\sigma_i = \frac{\sigma_o}{f_i}, z = N(0, \sigma_i) \tag{13}$$

p_{m0}: Minimum binary mutation probability
σ_o: Minimum mutation standard deviation
σ_i: Local mutation standard deviation
f_i: Local Fitness value
z: Normal random variable

3.4. Evolution parameters

Once genetic operators are designed, we have to tune the reference evolution parameters before the optimization. Evolution parameters are:

- λ_o: Reference selection pressure
- pc_o: Reference crossover probability
- pm_o: Reference mutation probability
- S: Size of population
- T: Maximum number of generations

We based the choice of evolution parameters on theory and experiments. For adaptive operators, it is very important the reference parameters make them stable and meaningful. So, we consider the actions of operators with these parameters at the beginning and at the end of the evolution. When fitness is low, operators favors exploration and the population is kept scattered. When fitness is high, the population should have converged near optimal solution (low diversity). These considerations based on non-adaptive boundary cases drive us to chose:

$$\lambda_o = 100, pc_o = 0.6, \sigma_o = 0.001,$$
$$pm_o = 0.0001, S = 20, T = 50$$

4. Simulation Results

We compared the algorithms for a 3D position/orientation with obstacles task. These algorithms are the TGA; the MEA and the Adaptive Multi-Chromosome Evolutionary Algorithm (AMEA).

We authorized up to 8 segments to allow redundant kinematics in order to avoid obstacles and/or singularities. Design parameters are the topology (16 link lengths) and the four joint configurations (no base positioning). The features of the algorithms are presented in Table III.

The task specifications are those described by the tetrahedron organization of section 2.1. Four goals have been defined on its vertices and six obstacles on the middle of its edges (see Table IV). The manipulator is evolved to reach these goals, while optimizing all performance criteria and avoiding obstacles. The security distance (or obstacles diameter) is set to 0.1 m and the mass unit of actuators (as well as linear mass of links) is set to 1 kg.

Figure 9 shows the evolution of the best fitness versus generations with the TGA, the MEA and the AMEA. The results have to be scaled because of the difference of time processing between the TGA (6 h) and the new algorithms (3 min) to perform the 20×50 evolution (SUN/SPARC5).

Table III. Three evolutionary algorithms.

Area	TGA	MEA	AMEA
Coding	all binary	binary/float	binary/float
Evolution	separate	global	global
Operators	blind	blind	adaptive
# Solutions	10^{116}	10^{211}	10^{211}
# Evaluations	100000	1000	5000
Computation time	6 h	3 min	3.5 min

https://doi.org/10.1017/S0263574707003931 Published online by Cambridge University Press

Table IV. Task specifications.

Object	X	Y	Z	Rx	Ry	Rz
Task 1	1	1	1	90	0	0
Task 2	2	1	1	0	180	0
Task 3	1.5	2	1	0	0	270
Task 4	1.5	1.5	2	20	90	45
Obstacle 1	1.5	1	1			
Obstacle 2	1.75	1.5	1			
Obstacle 3	1.25	1.5	1			
Obstacle 4	1.25	1.25	1.5			
Obstacle 5	1.75	1.25	1.5			
Obstacle 6	1.5	1.75	1.5			

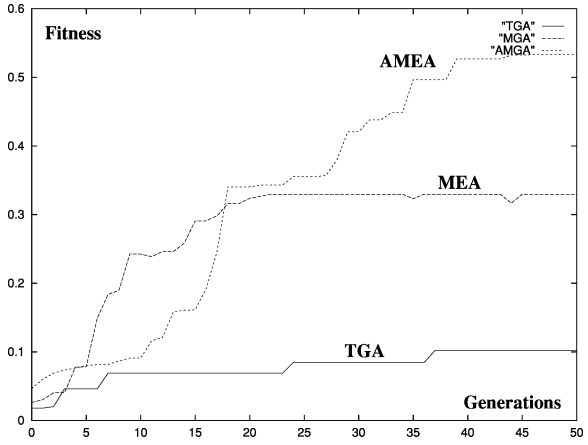

Fig. 9. Best fitness comparison.

The reason comes from MEA and AMEA do only one IKP evaluation for each topology while TGA does one full IKP evolution. It clearly appears that not only the multi-chromosome algorithms are, at least, 100 times quicker, but also that the optimized best solution is far better (more significant for AMEA).

Figure 10 shows the mean population fitness for the TGA, MEA, and AMEA. The differences here are more striking as we see that the population is much more improved with MEA and AMEA. Moreover, it is suggested the adaptive algorithm continues the improvement of the whole population as long as the best fitness is not reached for all (no flat in the AMEA

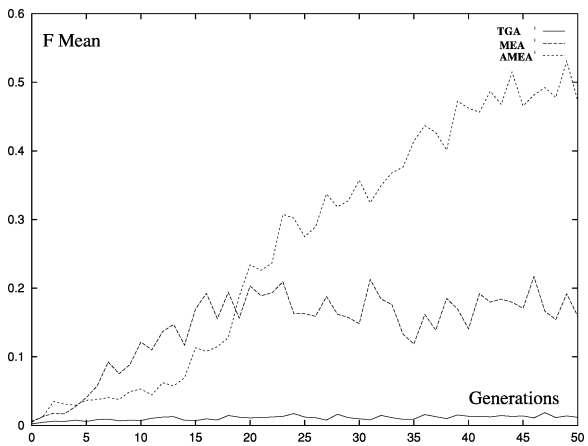

Fig. 10. Mean fitness comparison.

Table V. Best manipulator design parameters.

DOF	1	2	3	4	5	6	7	8
α	Ry	Rz	Ry	Rz	Rx	Rz	Rz	Ry
λ	R	P	R	R	R	F	R	R
l	12	2	8	3	12	5	8	1

Table VI. Satisfaction of criteria for best manipulator.

Criteria	L	A	I	O	D
k_i	0.5	1	2	0.5	1
Best	0.125	0.114	0.404	0	0
Performance	5 *cm*	18°	21.8 kg	max	max

evolution). TGA and MEA will always keep a constant gap between best and mean fitness because of the constant mutation rate while the AMEA will decrease the mutation noise with increasing fitness allowing a more complete convergence. This is similar to the *Simulated Annealing* temperature that decreases with time, allowing less and less variations and eventually drives all individuals toward a single solution.[24]

The best solution is a redundant manipulator (7DOF) and design parameters (Table V) include joint relative orientations α, joint types λ and link lengths l. We can see that the last link is much smaller than the others. This solution tends toward the particular design of a concurrent axis wrist that is known to be very helpful.

Criteria satisfaction for this manipulator are given in Table VI. Constraints are verified since all subtask configurations are out of obstacles ($O = 0$) and without kinematic singularities ($D = 0$). The primary objectives are not completely achieved but near enough to hope a fully valid pose after small joint adjustments. This have been done using ACT robot design software (by Aleph Technologies) for subtasks matching from evolved configurations (Fig. 11). Finally, the preference I has been respected well enough considering the distance of subtasks.

5. Conclusions

The algorithms we presented are progressive adaptations of evolutionary computations to task-based robot synthesis. They include a representation based upon a 3D modular kinematics. The multiobjective optimization is quickly performed with regard to the total search space and the complex problem. The TGA is a simple approach of the synthesis problem since it considers the IKP as an uncoupled subproblem. By including knowledge of the IKP inference MEA, and the actions of adaptive operators in AMEA, we introduced important bias in the search, improving the optimization. We used no database or robotic design experience to give rise to solutions from random seed. Evolved topologies have rather good performance criteria and represent some making sense options for this kind of design. The method is very flexible and adaptable as we can chose through the fitness function any criteria to optimize whatever its form and relations with the encoded solutions. It is important to notice that the whole manipulator and its configurations have been optimized in parallel. This

https://doi.org/10.1017/S0263574707003931 Published online by Cambridge University Press

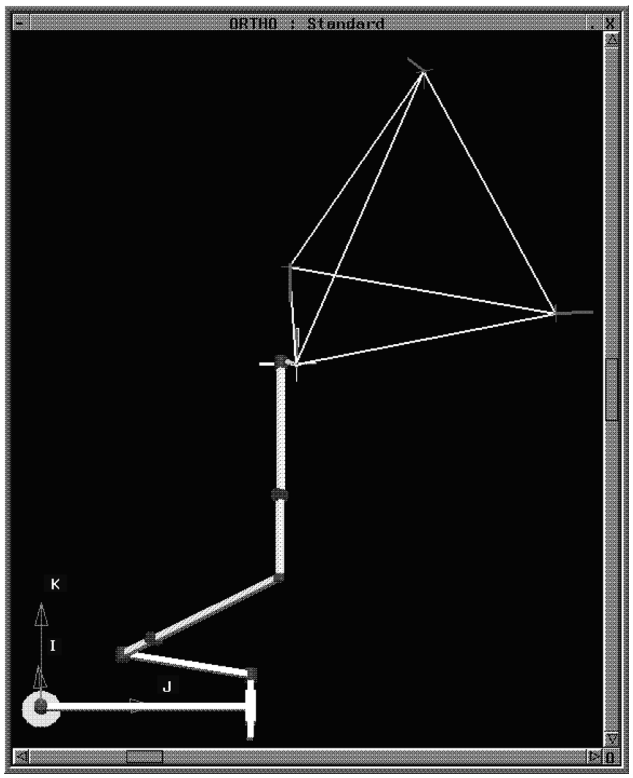

Fig. 11. A 7-DOF manipulator genetically evolved.

guarantees a global optimization of both structure (topology) and behavior (configurations). Here, the behavior were reduced to joint static values, but it can easily move to control systems which could be evaluated through a dynamic simulation for complex tasks.

Since technical auto reconfigurable solutions has been proposed and proven,[25] it is obvious than evolutionary optimization may be applied to these systems. Meanwhile, the main difficulty lies in the control distribution among the modules. The evolutionary process may work on the global system with a better result than locally because of its own qualities. Integrating the global decision process of reconfiguration into distributed modular systems is still a challenge for researchers throughout the world as outlined recently 26.

References

1. T. Fukuda and S. Nakagawa, "Dynamically reconfigurable robotic system," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Philadelphia, PA, USA (1988) pp. 1581–1586.
2. C. Paredis and P. Khosla, "A rapidly deployable manipulator system," *IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, MN, USA (1996) pp. 1434–1439.
3. M. Yim, "A reconfigurable modular robot with many modes of locomotion," *Proceedings of International Conference on Advanced Mechatronics*, Tokyo, Japan (1993) pp. 283–288.
4. D. Rus and G. S. Chirikjian, "Self-reconfigurable robots," *Auton Rob* **10**(1), 5–5 (2001).
5. R. O. Ambrose, Design, construction and demonstration of modular, reconfigurable robots *Ph.D. Thesis* (Austin, USA, University of Texas, 1991).
6. I. M. Chen and J. Burdick, "Determining task optimal modular robot assembly configurations," *IEEE International Conference on Robotics and Automation (ICRA)*, Nagoya, Japan (May 1995) pp. 132–137.
7. S. Hornby, H. Lipson and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *IEEE Trans. Robot. Automat.*, 19(4), 703–719 (2003).
8. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization- Algorithms and Complexity* (Prentice-Hall, Englewoods Cliffs, NJ, USA, 1982).
9. J. S. Arora, O. A. Elwakeil and A. I. Chahande, "Global optimization methods for engineering applications: a review." *J. Str. Opt.* 9(3-4), 137–159 (1995).
10. S. N. Shen, M. Chew and G. F. Issa, "Kinematic structural synthesis of mechanisms using knowledge-based systems," *J. Mech. Des.* **117**, 96–10 (1995)].
11. J. O. Kim and P. Khosla, "A multi-population genetic algorithm and its application to design of manipulators," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* Raleigh, NC, USA (1992) pp. 279–286.
12. C. Paredis and P. Khosla, Kinematic design of serial link manipulator from task specification. *International Journal of Robotics Research*, **12**(3), 274–287, (1993).
13. A. A. Khwaja, M. O. Rahman and M. G. Wagner, "Inverse kinematics of arbitrary robotic manipulators using genetic algorithms," **In**: Advances in Robot Kinematics: Analysis and Control. (J. Lenarcic and M. L. Husty, eds., (Kluwer Academic Publishers, 1998).
14. G. Yang and I.-M. Chen. Task-Based Optimization of Modular Robot Configurations—MDOF Approach. *Mechanism and Machine Theory*, 35(4), 517–540 (2000).
15. P. Chedmail and E. Ramstein, "Robot mechanism synthesis and genetic algorithms," *IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, MN, USA (April, 1996) pp. 3466–3471.
16. J. Pollack, H. Lipson, S. Ficici, P. Funes, G. Hornby and R. Watson. "Evolutionary techniques in physical robotics," Proceedings of the Third International Conference on Evolvable Systems, Edinburgh, UK (April, 2000) pp. 175–186.
17. O. Chocron and Ph. Bidaud, "Genetic design of 3d modular manipulators," *IEEE International Conference on Robotics and Automation (ICRA)*, Albuquerque, NM, USA (April 1997) pp. 223–228.
18. O. Chocron and Ph. Bidaud, "Evolutionnary algorithms in kinematic design of robotic systems," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Grenoble, France (September 1997) pp. 1111–1117.
19. F. Chapelle, O. Chocron and Ph. Bidaud, "Genetic programming for inverse kinematics problem approximation," *International Symposium on Robotics (ISR)*, Montreal, Canada (May 2000) pp. 5–11.
20. S. Sakka and O. Chocron, "Optimal design, configurations and positions for a mobile manipulation task using genetic algorithms," *TENTH IEEE International Conference on Robot and Human Communication (ROMAN)*, Paris-Bordeaux, France (September 2001) pp. 268–273.
21. T. Yoshikawa. *Foundations of robotics: analysis and control* (MIT Press Cambridge, MA, USA, 1990).
22. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989).
23. T. Bäck, *Evolutionary Algorithms in Theory and Practise* (Oxford University Press, New York, NY, USA 1996).
24. A. C. Thornton, "Genetic algorithms versus simulated annealing: satisfaction of large sets of algebraic mechanical design constraints," *Proceedings of Artificial Intelligence in Design*, Lausanne, Switzerland (August 1994) pp. 381–400.
25. E. Yoshida, S. Murata, K. Tomita, H. Kurokawa and S. Kokaji. Distributed formation control for a modular mechanical system. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Grenoble, France (1997) pp. 1090–1097.
26. M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems—challenges and opportunities for the fiture," *IEEE Robot. Automat. Mag.* **14**(1), 43–52 (2007).

https://doi.org/10.1017/S0263574707003931 Published online by Cambridge University Press