Evolutionary Cost-Optimal Composition Synthesis of Modular Robots Considering a Given Task

Esra Icer[1,3], Heba A. Hassan[2,3], Khaled El-Ayat[2] and Matthias Althoff[1]

Abstract— **Commercially available robots cannot always be adapted to arbitrary tasks or environments, particularly when the task would exceed the kinematic or dynamic limits of the robot. Modular robots offer a solution to this problem, since they can be reconfigured in various ways from a set of modules. The challenge of choosing the optimal composition for a given task, however, is hard since the search space of compositions is vast. Our approach addresses this problem: instead of finding the cost-optimal solution over all possible compositions individually, we propose a time-efficient composition synthesis method which uses evolutionary algorithms by taking task-related objectives into account. Simulations show that our algorithm finds the cost-optimal module composition with less computation time than other methods in the literature.**

I. INTRODUCTION

One of the current trends in robotics is to find new solutions for industrial robots for different tasks or changing environments. Although the available industrial robots offer high efficiency, robustness, and accuracy for specific tasks, it is not easy to adapt them for different tasks or environments. As a solution to this problem, modular robot manipulators consisting of several interchangeable, pre-designed modules which enable one to configure various manipulators for different tasks are proposed [1]. The unique properties of modular robots, such as easy modification, easy maintenance, flexibility and high versatility, make them a promising technology for future flexible manufacturing scenarios. Since the structure of the robot is different for each module combination, modular robot synthesis is a complex and time-consuming combinatorial problem. In this work, we propose an algorithm for selecting the cost-optimal composition from a set of given modules for a given task. We define the *task* as carrying a given payload from an initial pose to a final pose passing through intermediate points in a cost-optimal way while avoiding all obstacles in the environment.

We review previous works on cost-optimal, task-based composition synthesis of modular robots considering a desired objective function. Both deterministic and stochastic methods have been used so far. In [2], an optimal composition synthesis method based on the genetic algorithm (GA) is proposed which compares different compositions based on an evaluation function to determine how well the composition fulfills a given task in an obstacle-free environment. Another GA-based optimal composition synthesis method for modular robots considering only the physical properties of modules is proposed in [3]. A simulated-annealing-based method which considers the kinematic constraints is presented in [4]. In [5], a two-level GA-based method for tasks in environments with static obstacles is proposed. This method mainly consists of a top-level GA to generate compositions from the given modules and a lower-level GA to optimize joint positions. The same authors present another technique which stores both composition and configuration information in the same chromosome [6]. In [7], a GA-based modular robot composition synthesis method for obtaining optimal kinematic configurations for task-based, fault-tolerant manipulators is proposed. A concurrent optimal design approach, which mainly depends on grouping the variables to reduce the number of independent variables in the optimization process, is proposed in [8]. In [9], the task requirements are split into 3 groups: *i*) those which must be satisfied, *ii*) those without rigid targets, and *iii*) requirements not only to be satisfied but also to be optimized. A progressive task-based design approach for non-modular manipulators is proposed in [10], which divides the design procedure into kinematic design, planning, and kinematic control. A GA-based multi-solution inverse kinematics solver is proposed in [11] to find the best inverse kinematics solution for modular robots. All the methods mentioned above only consider the given poses in the task definition.

In contrast, we proposed a composition synthesis method based on the elimination of unfeasible compositions in our previous work [1], which not only takes the initial and goal positions into account but also considers whether the given task is achievable. We have also presented a cost-optimal composition synthesis method which is mainly based on the elimination of less likely compositions during the optimization process [12].

However, although obstacles are not considered in most work for modular robots, it is important to consider how paths are planned in an environment with obstacles. Studies on collision-free path planning are mainly based on two approaches: *i*) configuration space (C-space) approaches [13], [14] and *ii*) task space approaches [15]–[17]. In C-space approaches, an n-dimensional space is constructed for a manipulator with n degrees of freedom (DOFs), and the robot is represented by a point [18]. In contrast to C-space approaches, task space approaches try to find a solution directly in the workspace of the robot [18]. Since our goal is to find collision-free paths for a large number of unique kinematic chains, the transformation of obstacles

[1] Department of Computer Science, Technische Universität München, 85748 Garching, Germany. {icer, althoff}@in.tum.de
[2] Department of Computer Science and Engineering, American University in Cairo, 113 Kasr Al Aini Street Cairo, Egypt. {heba.ali, kelayat}@aucegypt.edu
[3] These authors contributed equally to this work.

for each composition from task space to joint space is time-consuming. As a solution to this problem, we check the collision in workspace although the path planning is either done in configuration space or task space; this offers simplicity and computational efficiency.

In this paper, we propose a composition synthesis algorithm for modular robots for finding the cost-optimal module composition using evolutionary algorithms. Compositions are varied by changing the modules and their combinations. In contrast to previous literature, we *i*) present a computationally-efficient composition synthesis algorithm for modular robots using evolutionary algorithms, *ii*) consider additional task-related objectives in the evaluation of the compositions and generate a set of the best compositions regarding the evaluation function, and *iii*) find an individual cost-optimal solution for the set of the best compositions (i.e. the optimal composition and its corresponding optimal trajectory).

This paper is organized as follows: Sec. II explains the combinatorial problem for modular robots. Sec. III describes our optimal-composition synthesis method, followed by implementation details in Sec. IV. Finally, conclusions are drawn in Sec. V.

II. Problem Statement

Throughout this paper, we consider modular and serially connected manipulators with n degrees of freedom, whose kinematics are uniquely determined by a vector $\mathbf{q} \in \mathbb{R}^n$ of joint positions, where $\mathbf{q}$ refers to angles for rotational joints and translations for prismatic joints. Each possible composition is referred to by $k \in \{1, \ldots, N\}$, where N is the maximum number of possible compositions without considering constraints specified by a given task. The task requirements are constrained by the kinematic model of the robot composition and the static obstacles in the environment. The environment, consisting of static obstacles and the robot, is indicated by $\mathcal{W} \subset \mathbb{R}^3$. The space occupied by the robot as a relation of its joint position vector is denoted by $\mathcal{A}(\mathbf{q}) \subset \mathcal{W}$. The joint vector of the k^{th} composition is denoted by $\mathbf{q}_k$, and the space occupied by the k^{th} composition is presented by $\mathcal{A}(\mathbf{q}_k) \subset \mathcal{W}$.

The obstacles are represented by arbitrary geometric shapes in $\mathbb{R}^3$; the occupancy of the j^{th} obstacle in the workspace is denoted by $\mathcal{O}_j \subset \mathcal{W}$, and the union of all obstacles is presented by $\mathcal{O} = \bigcup_j \mathcal{O}_j$. The obstacle-free space in the environment is defined as $\mathcal{F} = \mathcal{W} \setminus \mathcal{O}$.

Time is indicated by $t \in [0, t_f]$, where t_f is the total time to reach the goal position and the function $\mathbf{q}(t)$ maps time t to the joint position vector. The forward kinematics of the robot with joint vector $\mathbf{q}(t)$, which represents the pose of the end effector, is represented by $f(\mathbf{q}(t))$. Considering the given task, it is assumed that all compositions start from a given initial position $\mathbf{p}_s$, defined as $\mathbf{p}_s = f(\mathbf{q}(0))$, and terminate at a given final position $\mathbf{p}_g$, defined as $\mathbf{p}_g = f(\mathbf{q}(t_f))$. An optimal path is generated between $\mathbf{p}_s$ and $\mathbf{p}_g$, which considers a desired objective function and a set of possible intermediate positions, defined as $\mathcal{P}_{int} = \{f(\mathbf{q}(t)) \mid \mathbf{q}(t) \in \mathcal{Q}_t\}$, where $\mathcal{Q}_t$ is the discrete set of intermediate joint values and $\mathcal{P}_{int} \subset \mathcal{F}$.

The variable η indicates the cost-optimal module composition which fulfills the given task by considering a given objective function $g(\cdot)$. It is described as

$$\eta = \underset{k \in \mathcal{K}}{argmin}\ g_k(\cdot), \tag{1}$$

where $\mathcal{K}$ is the set of modules that can achieve the given task, which is defined as

$$\begin{aligned} \mathcal{K} = \{k \mid\ & 1 \leq k \leq N \wedge \forall t \in [0, t_f]\ \exists\ \mathbf{q}_k(\cdot) : \\ & \mathcal{A}(\mathbf{q}_k(t)) \cap \mathcal{O} = \emptyset \\ & \wedge\ \ \mathbf{q}_k(t) \in\ [\mathbf{q}_{k,min},\ \mathbf{q}_{k,max}] \\ & \wedge f(\mathbf{q}_k(t)) \in \mathcal{P}_{int} \\ & \wedge f(\mathbf{q}_k(0)) = \mathbf{p}_s\ \wedge f(\mathbf{q}_k(t_f)) = \mathbf{p}_g\}. \end{aligned} \tag{2}$$

In the next section, we describe the solution concept for the given problem statement.

III. Proposed Method

A large number of different possible compositions can be generated by varying modules and by changing their order. This large design space makes modular robot synthesis complex and time-consuming. In our previous work [1], we proposed a composition synthesis method for obtaining all feasible compositions for a given task. That method mainly generates all possible compositions using a brute-force algorithm and aims to reduce the computation time by testing feasibility with increasingly complex tests. First, it checks only whether the compositions are feasible at the initial and goal positions. Then, it checks whether a path exists for each remaining composition. Finally, the task requirements are considered and individual solutions for each composition are found (see Fig. 1(a)). Moreover, we have also introduced a method for efficiently obtaining the cost-optimal solution while eliminating the cost-inefficient compositions during the optimization process [12]. In contrast to our previous papers, we propose a cost-optimal composition synthesis method for modular robots using evolutionary algorithms instead of our previous enumeration algorithm.

Throughout the paper, we assume that *i*) four different types of modules are considered: *bases*, *joints*, *links*, and *end effectors*; *ii*) all modules have only one input and only one output port; and *iii*) there is only one base module and one end effector module in the robot structure. We consider 1-DOF joint modules, y-DOF end effector modules where $y \in \mathbb{N}$ and 0-DOF link and base modules in the modular robot structure. To restrict the search space, we use the following pre-determined structure:

Base - Joint - Link - Joint - · · · · - Link - End Effector.

In order to avoid iterating over all possible compositions and to check whether a reachable path exists for each composition, we propose a GA-based synthesis algorithm in this work. The schematic representation of the proposed method is shown in Fig. 1(b). The method consists of three sequential

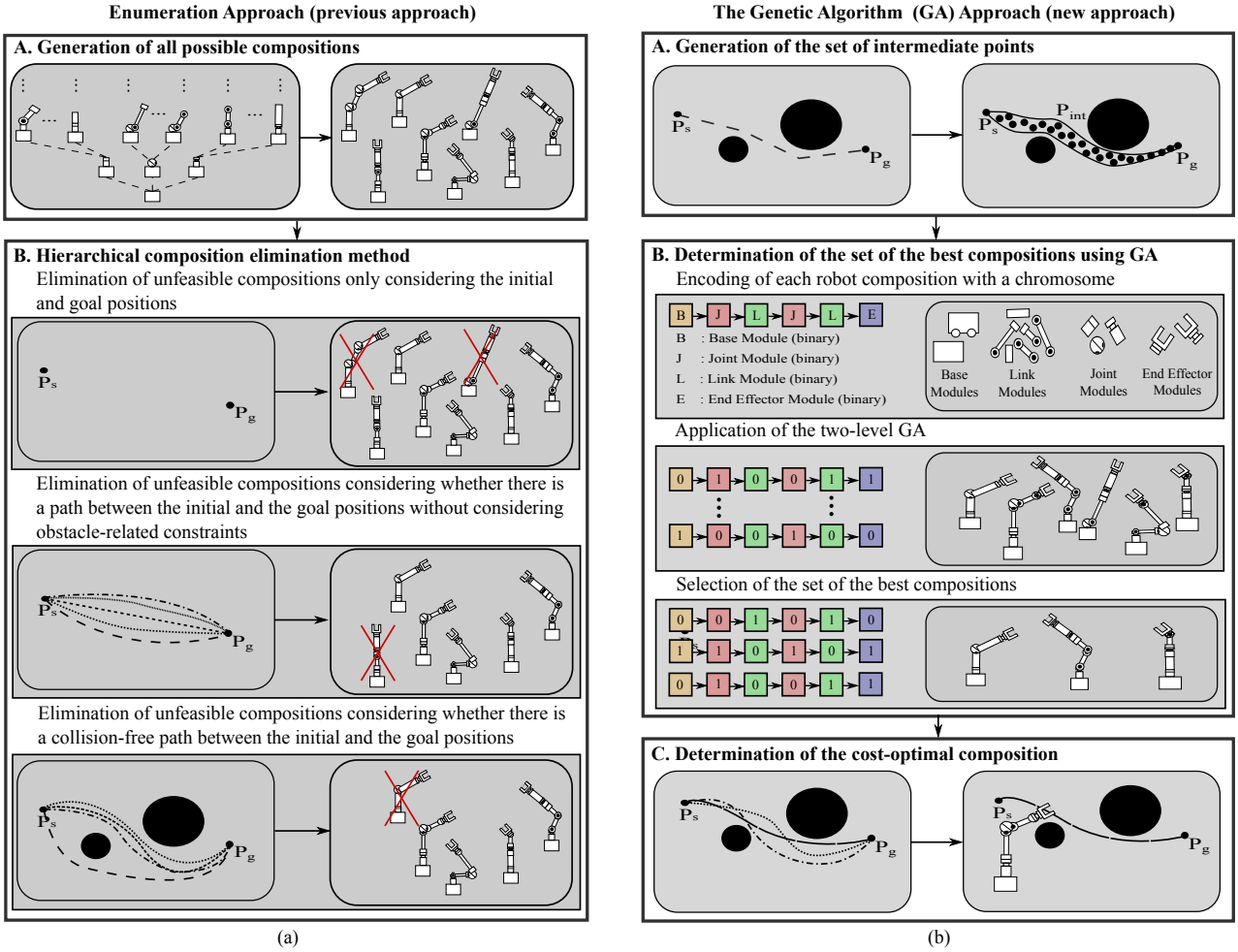

Fig. 1. The composition synthesis methods for modular robots: (a) the previously proposed feasible composition generation method in [1] and (b) the proposed cost-optimal composition synthesis method.

steps: *A*) generation of a set of intermediate points along the cost-optimal path from the initial position to the goal position only considering the position of the end effector, *B*) determination of the set of the best compositions using GA considering additional task-related objectives which are obtained in (A), and *C*) determination of the cost-optimal composition considering the given objective function for the selected compositions from (B).

A. Generation of the set of intermediate points

The first step of the proposed algorithm is the generation of a set of intermediate points between the initial and goal positions considering the task requirements. To determine these points, the shortest geometric path is generated only taking the end effector's position into account. Any geometric path planning technique can be used to find the shortest path for which the end effector avoids collisions. In this work, we use the Wavefront algorithm which is illustrated in Fig. 2 to determine these points due to its simplicity [18]. Following the Wavefront algorithm, the $3D$ environment is divided into *cells* with a certain *cell size*, and all *cells* are initially numbered with 0. Then, cells containing obstacles or the goal point are numbered with 1 or 2, respectively. The rest of the cells are numbered considering the distance from the goal position to each cell as in Fig. 2(a). Although we implement the algorithm in $3D$, it is shown in $2D$ in Fig. 2 for illustration purposes. The obtained paths follow values in an increasing order which start from the goal position and finish at the initial position. The yellow regions in Fig. 2(b) show the set of intermediate points to be followed by the end effector to fulfill the task with the shortest path.

Fig. 2. Implementation of the Wavefront algorithm: (a) numeration of cells, (b) the set of intermediate points, and (c) the unfeasible points and the optimal path obtained by the Wavefront algorithm.

B. Determining the set of the best compositions using GA

To determine the set of the best compositions considering the task-related objectives, we use a GA since it is suitable for our problem which is multidimensional, nonlinear, and has a large search space. GAs are based on the concept of *chromosomes*, which is the encoding of a solution to the problem at hand. A *chromosome* is a sequence of *genes* which store data about each module. The *population size* is the number of chromosomes in one generation and the *generation size* is the maximum number of iterations for the GA. In our particular problem, *chromosomes* represent robot compositions and they are represented in binary (genes are 0 and 1 as presented later in Sec. IV) as in Fig. 1(b). To evaluate each individual *chromosome*, we apply three GA operators (*crossover*, *mutation*, and *selection*) between one generation and the following one [19]. *Crossover* is done by selecting two parents to produce an offspring. *Mutation* is a way to increase the diversity of a population since new traits are introduced into the genome, thus allowing the population to explore the large solution space. *Selection* is used to improve exploitation by passing the best individuals from one generation to the next one [20].

To decrease the search space, we implement the GA in two levels. In the first level, we quickly remove compositions which are not able to reach the initial and goal positions. To do that, we apply a kinematic filter, a static torque-force filter, and a collision filter as in [1]. In the second level of the GA, compositions which are able to pass the previously mentioned tests are evaluated based on the following criteria:

1) *reachability* (R) - the distance between the task point and the base of the robot d_{d2b} calculated by (3), where l_i is the length of the i^{th} module in the composition and n is the number of DOFs.

$$R = \frac{d_{d2b} - \sum_{i=1}^{2n} l_i}{d_{d2b}} \tag{3}$$

2) *linear distance* (L) - the normalized distance between the position of the end effector $\mathbf{p}_e$ and the position of the desired point $\mathbf{p}_d$.

$$L = \frac{\|\mathbf{p}_d - \mathbf{p}_e\|}{\|\mathbf{p}_d\|} \tag{4}$$

3) *angular distance* (A) - the normalized distance between the orientation of the end effector $\mathbf{r}_e$ and the orientation of the desired point $\mathbf{r}_d$.

$$A = \frac{\|\mathbf{r}_d - \mathbf{r}_e\|}{\|\mathbf{r}_d\|} \tag{5}$$

4) *dexterity* (D) - the ability to move and apply forces in arbitrary directions as easily as possible. It is obtained by (6), where J is the Jacobian matrix and w is the Yoshikawa manipulability index calculated by $w = \sqrt{det(JJ^T)}$ [21].

$$D = \frac{1}{1+w} \tag{6}$$

5) *involved modules* (I) - the measure of the mass and structure complexity of the robot calculated by (7), where l_i is the length of the i^{th} module in the composition, ζ is the joint type which is 1 for prismatic joints and 0 for revolute joints, $l_{e,i}$ is the maximum limit of the i^{th} joint, and d_{e2b} is the distance between the end effector and the base.

$$I = \frac{\sum_{i=1}^{n} l_i + \zeta \cdot l_{e,i}}{d_{e2b}} \tag{7}$$

6) *joint value differences between the initial and goal positions* (V) - the sum of the differences between the value of each joint

$$V = \sum_{i=1}^{n} |q_i(0) - q_i(t_f)|. \tag{8}$$

The compositions with smaller V values are better solutions for our particular problem.

7) *obstacle proximity* (O) - the minimum distance between the obstacles and each robot component which is calculated by (9), where $d_{c_i 2 O_j}$ represents the distance between the i^{th} component and the j^{th} obstacle, r_{O_j} is the radius of the i^{th} obstacle, r_{c_i} is the radius of the i^{th} component, and s is the total number of obstacles in the environment (for more details see [1]).

$$O = \underset{i \in 1 \ldots 2n,\ j \in 1 \ldots s}{argmin} \left(d_{c_i 2 O_j} - (r_{O_j} + r_{c_i}) \right) \tag{9}$$

The criteria R, L, A, D, and I are also considered in [6] and O is modified from [6] to obtain less conservative distances. One additional criterion, V, is added to these evaluation functions to minimize and optimize differences between joint angles. We combine all evaluation criteria to obtain the objective function

$$g = w_1 \cdot e^{-(k_1 \cdot R + k_2 \cdot L + k_3 \cdot A + k_4 \cdot D + k_5 \cdot I + k_6 \cdot V + k_7 \cdot \frac{1}{1+O})} + w_2 \cdot P \tag{10}$$

where k_i are the weighting values for each criteria, w_i are the weighting values of the objective function, and P is the percentage of the intermediate points that a composition can reach. The first part of the equation represents the method proposed in [6]. Adding the second part of (10) enables us to select the compositions considering the task-related objectives. A set of the best individuals reached by the end of the simulation are passed on to the next step as explained in Sec. III-C.

C. Determination of the cost-optimal composition

After finding the set of the best compositions using the GA, we aim at finding an individual collision-free path between the initial and goal positions for each of them considering the whole robot. To find an optimal path, any path planning algorithms can be used. In this work, we use two different algorithms: *i*) the Wavefront algorithm and *ii*) the rapidly exploring random trees (RRTs) algorithm [22]. The Wavefront algorithm is a deterministic approach which always gives the same result; in contrast, the RRT is a stochastic approach which gives different results for

each run. The reasons why we choose these algorithms are *i*) they do not get stuck in local minima, *ii*) they are easy to implement, and *iii*) they are computationally efficient for the considered problem.

We implement the Wavefront algorithm in task space (as in Sec. III-A) since the mapping of obstacles from task space to C-space for each composition is complex and computationally expensive, especially for high-DOF robots. However, since path planning is done in task-space, some regions cannot be reached by a composition. Red cells in Fig. 2(c) show the unreachable and torque-violating points which are numbered by 1. The final path goes from the goal point to the initial point, following increasing numbers as shown in Fig. 2(c).

We implement the RRT algorithm in C-space as a second method and check collisions in task space. Based on the method in [23], we generate two different trees: one of them starts from $\mathbf{q}_s$ and the other one starts from $\mathbf{q}_g$. A path is generated when these two trees intersect. More details about the implemented RRT algorithm can be found in [23].

After finding the cost-optimal solutions for each composition from both path planning methods, we compare all results and select the composition with the minimum cost value as in (1).

IV. Numerical Experiments

To demonstrate our synthesis algorithm, we have implemented it in MATLAB R2016b running on an Intel® Core™ i5 processor with 1.6 GHz and 4 GB of memory. We use one type of fixed base module (coded as 0), two types of one-DOF joint modules (coded as 0 and 1), three types of zero-DOF link modules (coded as 00, 01, and 10), and two types of one-DOF end effector modules (coded as 0 and 1), which are shown in Fig. 3. The properties of each module are given in Tab. I. Throughout the simulations, we consider the robot structure defined in Sec. III. The base module is positioned at point $\mathbf{p}_b = (0,\ 0,\ 0)^T$. In this example, we only consider 6-DOF robots which means we consider 15552

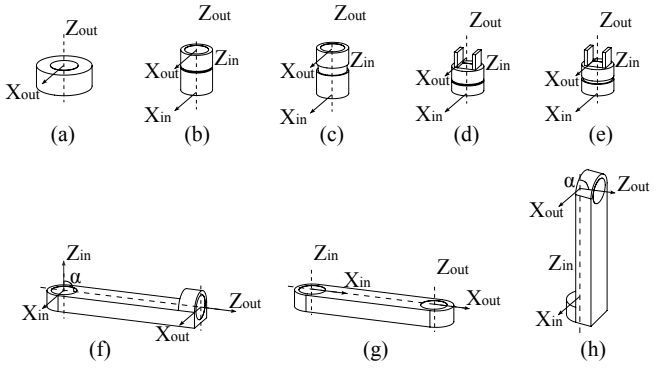

Fig. 3. Basic modules used in this study: (a) base module; (b) rotational joint module; (c) prismatic joint module; (d) rotational end effector module; (e) prismatic end effector module; (f) $\alpha = 90^\circ$ link module, whose length runs along the y direction of the previous coordinate system; (g) $\alpha = 0^\circ$ link module and (h) $\alpha = -90^\circ$ link module, whose length runs along the z direction of the previous coordinate system.

TABLE I

The module parameters involved in simulation (see Fig.3)

	Length [m]	Diameter [m]	Max τ [Nm]	Joint Limits [rad or m]
L1	0.75	0.2	-	-
L2	0.75	0.2	-	-
L3	0.75	0.2	-	-
J1	0.25	0.2	80	[-π,π]
J2	0.25	0.2	75	[0,0.2]
EE1	0.2	0.2	75	[-π,π]
EE2	0.2	0.2	70	[0,0.1]

different robot compositions. The task is defined as carrying a 5 kg payload from the initial position $\mathbf{p}_s$ to the goal position $\mathbf{p}_g$ along the shortest path (which refers to the variable g in (1)) without colliding with the obstacles in the environment and without violating the joint limits. To test our algorithm, we restrict the environment by a cube of 7 m edge length and center it at $\mathbf{p}_b$. The obstacles are defined as spheres and their radii are randomly generated within $r_o = [0.05, 0.4]$. We randomly generate 20 different scenarios within the aforementioned environment limits, and the number of the obstacles for each scenario varies between 1 to 6.

We use a *step size* of 0.1 for the Wavefront algorithm and compute the inverse kinematics up to an accuracy such that the end effector remains within the corresponding *cell*. Then, we test *population sizes* and *generation sizes* while fixing *mutation* rate, *selection* rate, and *crossover* rate as shown in Tab. II (different values in group I and II are motivated later) since the search space is huge. The *population size* and the *generation size* are closely related and complement each other in terms of the exploration and diversification of the solutions. Experiments are done in order to determine the optimal *population size* and *generation size* by comparing the obtained best fitness value as well as the required computation time. We test values for different scenarios and all of them provide similar results. Outcomes of different *population sizes* and *generation sizes* for scenario 1 are illustrated in Fig. 4, which shows that the best fitness value remains the same (see Fig. 4(b)) although the simulation time increases (see Fig. 4(a)). Based on the results in Fig. 4, we choose the *population size* as 100 and the *generation size* as 150 (see Tab. II).

During the execution of the GA, we use two groups of values (see Tab. II). Initially, we use group I to prioritize the

TABLE II

The GA Parameters

	Group I	Group II
the *population size*	100	
the *generation size*	150	
the *mutation* rate	60%	30%
the *selection* rate	10%	10%
the *crossover* rate	30%	60%

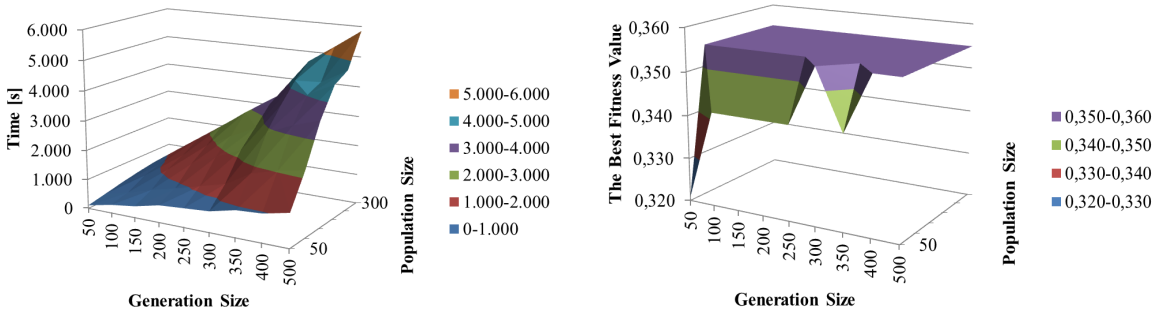

Fig. 4. The simulation results for different population size and generation size: (a) simulation time and (b) the best fitness value for each simulation.

diversity of the population in the first few generations while setting high *mutation* rate. This is so that the algorithm does not get stuck in local minima. After a few generations we use values of group II to give more importance to the best individuals to increase exploitation.

While defining GA parameters, we give the same priority to each criterion in the first part of (10) described in Sec. III-B and set each k_i as $1/7$. We also give the same priority $w_1 = 0.5$ and $w_2 = 0.5$ in (10) and consider the best 20 individuals for each scenario which are reached in the last generation of the GA and call it *the set of the best compositions*. Then, paths are generated for the best compositions using the methods detailed in Sec. III-C to show that the effectiveness of the proposed algorithm is independent of the used path planning method.

Simulation results for each scenario are given in Tab. III, where the success ratio (sr) is the ratio of the number of compositions that can find a feasible path to the number of *sets of the best compositions*. Simulations show that 73% of the compositions (on average) are able to find feasible paths for the proposed method (PM) using the Wavefront (WF) algorithm (see Tab. III). As mentioned before, we also use a RRT algorithm where random points are generated in C-space with a *maximum step size* of 0.1 in order to be consistent with the WF algorithm, and the *maximum number of points* is set to 100. Since RRT gives different results for each run, we run the path planning algorithm 5 times per composition and consider the average for each composition. Simulations show that 96% of the set is able to find a collision-free path on average (see Tab. III).

We compare our proposed method with the method in [6]. To make a fair comparison, we also select *sets of the best compositions* obtained from the algorithm and implement the same path planning methods with exactly the same parameters. We compare the results obtained from the WF algorithm and on average only 48% of the sets can find collision-free paths. We also run the RRT algorithm and 91% of the set is able to find a collision-free path on average. For 2 scenarios, the best compositions obtained from [6] cannot find a solution since the robots collide with the obstacles in the environment while performing the task. The simulations show that although the computational time of [6] is shorter than that of the proposed method, the best compositions only considering the initial and goal positions are not the cost-optimal composition when the full task is taken into account.

We also compare the proposed method with our previous method in [1] and compare the total computation time. To make a better comparison, we implement the same path planning algorithms on the remaining compositions, calculate the ratio of the simulation time for each scenario, and show the ratio of the computation time denoted by tr in Tab. III. Total simulation time for both path planning methods are also given in Tab. III. The simulations show that the proposed algorithm is computationally more efficient in case there are multiple solutions for the given task. The computational efficiency increases with with the number of

TABLE III
EXPERIMENTAL RESULTS FOR EACH SCENARIO

	NR		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
PM	WF	sr	0.75	0.24	0.24	1	-	0.75	0.65	0.55	0.95	0.8	0.85	1	0.9	1	0.62	0.7	0.64	0.63	1	0.6
		tr	0,72	0.72	1.57	0.79	3.32	0.66	0.44	0.96	0.55	0.67	0.98	0.84	0.54	0.96	0.81	0.48	0.79	0.54	0.47	0.58
	RRT	sr	0.8	0.85	0.89	1	-	1	1	0.9	1	0.9	0.95	1	1	1	1	1	1	1	1	1
		tr	0.85	0.9	0.81	1.05	3.37	0.79	0.29	0.81	0.48	0.66	0.47	0.4	0.37	0.91	0.54	0.78	0.69	0.65	0.28	0.37
[6]	WF	sr	0.3	0.25	0.1	0.6	-	0.5	0.5	0.4	0.5	0.6	0.7	0.45	0.6	0.85	0.6	0.6	0.2	0.2	0.65	0.55
		tr	0.29	0.38	0.72	0.36	2.94	0.51	0.41	0.33	0.3	0.39	0.51	0.32	0.39	0.47	0.35	0.19	0.35	0.37	0.4	0.4
	RRT	sr	0.75	0.8	0.85	1	-	1	1	0.8	0.8	0.85	0.95	1	0.9	1	0.8	1	0.9	1	1	0.85
		tr	0.34	0.97	0.51	0.64	2.99	1.4	0.27	0.67	0.6	0.43	0.3	0.34	0.3	0.87	0.46	0.43	0.82	0.55	0.33	0.44
[1]	WF	t [h]	1.38	1.13	0.77	1.2	0.28	0.6	1.36	1.32	1.38	1.18	1	3.56	1.98	1.17	1.39	2.46	1.38	1.38	1.18	1.22
	RRT	t [h]	3.67	1.69	2.86	1.38	0.27	1.18	5.82	2.89	3.32	3.43	3.27	3.56	5.64	1.56	3.64	3.54	2.5	2.4	4.08	4.68

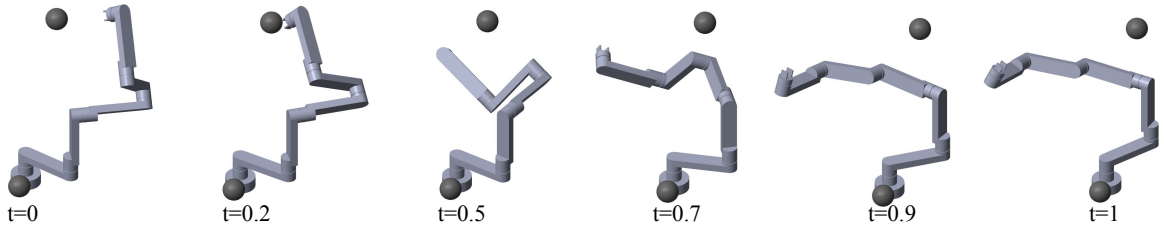

Fig. 5. Screen shots for the composition B/J1/L2/J1/L1/J1/L3/J1/L2/J1/L3/EE1 for the task defined as follows: $\mathbf{p}_s = (1.51.52)$, $\mathbf{p}_g = (-2 - 1.20.6)$, $\mathbf{p}_{o,1} = (1.5, -1.3, 1.1)$, $r_{o,1} = 0.34$ and $\mathbf{p}_{o,2} = (-1.2, 2.1, 0.8)$, $r_{o,1} = 0.39$ considering t_f=1

feasible compositions of the initial and goal positions.

V. Conclusion

In this paper, we present a task-based optimal composition synthesis method for modular and reconfigurable robot manipulators. To the best knowledge of the authors, none of the evolutionary algorithm-based approaches presented in the literature so far has considered task-related constraints in the evaluation function. This idea enables the user to generate the optimal composition of the modules in a time-efficient way, making modular and reconfigurable robots a more promising technology in the industrial environment. The comparisons show that our method is more efficient than the current method, which is based on eliminating compositions from the brute-force algorithm in case there are many compositions which can fulfill the task when only considering the initial and goal positions. It also shows that compositions which give the best fitness values only considering the initial and the goal positions do not provide the cost-optimal solution for the execution of the given task. The main advantages of the proposed optimal composition synthesis algorithm are *i*) it is applicable to all kinds of modules; *ii*) it only generates cost-optimal obstacle-free paths for the set of the best compositions, which makes it computationally efficient; and *iii*) it provides a faster solution when compared to finding assemblies by optimizing trajectories for each assembly individually.

Acknowledgment

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme FP7/2007- 2013/ under REA grant agreement number 608022 and from the American University In Cairo, Egypt.

References

[1] E. Icer, A. Giusti, and M. Althoff, "A task-driven algorithm for configuration synthesis of modular robots," in *Proc. IEEE International Conference on Robotics and Automation*, 2016, pp. 5203–5209.

[2] I. Chen and J. Burdick, "Determining task optimal modular robot assembly configurations," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1995, pp. 132–137.

[3] S. Farritor, S. Dubowsky, N. Rutman, and J. Cole, "A systems-level modular design approach to field robotics," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 1996, pp. 2890–2895.

[4] C. J. Paredis and P. Khosla, "Synthesis methodology for task based reconfiguration of modular manipulator systems," in *Proc. of the 6th International Symposium on Robotics Research, ISRR*, 1993.

[5] O. Chocron and P. Bidaud, "Evolutionary algorithms in kinematic design of robotic systems," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 1997, pp. 1111–1117.

[6] O. Chocron and P. Bidaud, "Genetic design of 3d modular manipulators," in *Proc. IEEE International Conference on Robotics and Automation*, 1997, pp. 223–228.

[7] Q. Li and J. Zhao, "A universal approach for configuration synthesis of reconfigurable robots based on fault tolerant indices," *Industrial Robot: An International Journal*, vol. 39, no. 1, pp. 69–78, 2012.

[8] Z. M. Bi and W. Zhang, "Concurrent optimal design of modular robotic configuration," *Journal of Robotic systems*, vol. 18, no. 2, pp. 77–87, 2001.

[9] W. Gao, H. Wang, Y. Jiang, and X. Pan, "Task-based configuration synthesis for modular robot," in *Proc. International Conference on Mechatronics and Automation (ICMA)*, 2012, pp. 789–794.

[10] J. Kim and P. K. Khosla, "Design of space shuttle tile servicing robot: an application of task based kinematic design," in *Proc. IEEE International Conference on Robotics and Automation*, 1993, pp. 867–874.

[11] S. Tabandeh, C. Clark, and W. Melek, "Task-based configuration optimization of modular and reconfigurable robots using a multi-solution inverse kinematics solver," in *International Conference on Changeable, Agile, Reconfigurable and Virtual Production, Toronto/Canada*, 2007.

[12] E. Icer and M. Althoff, "Cost-optimal composition synthesis for modular robots," in *Proc. IEEE Multi-Conference on Systems and Control*, 2016, pp. 1408–1413.

[13] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[14] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 1, no. 5, pp. 90–98, 1986.

[15] G. Mesesan, E. Icer, and M. Althoff, "Hierarchical genetic path planner for highly redundant manipulators," in *Proc. of the Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, 2015.

[16] J. Yu and P. Müller, "An on-line cartesian space obstacle avoidance scheme for robot arms," *Mathematics and Computers in Simulation*, vol. 41, no. 5, pp. 627–637, 1996.

[17] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space Voronoi bias," in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 2061–2067.

[18] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[19] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[20] S. Rylander and G. B., "Optimal population size and the genetic algorithm," *Population*, vol. 100, no. 400, p. 900, 2002.

[21] T. Yoshikawa, *Foundations of robotics: analysis and control*. MIT press, 1990.

[22] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[23] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.