Using Standard Components in Evolutionary Robotics to Produce an Inexpensive Robot Arm

Michael W. Louwrens, Mathys C. du Plessis and Jean H. Greyling

Abstract Evolutionary Robotics (ER) makes use of evolutionary algorithms to evolve controllers and morphologies of robots. Despite successful demonstrations in laboratory experiments, ER has not been widely adopted by industry as means of robot design. A possible reason for this is that current ER approaches ignore issues that are important when designing robots for practical use. For example, the availability and cost of components used for robot construction should be considered. A robot designed by the ER process may require specialised custom components to be built to support the physical functioning of the design, if the components selected by an ER process are not widely available. Alternatively, the ER designed robot may be too expensive to be constructed. This paper demonstrates that standard off-the-shelf components can be used by the ER process to design a robot. A robot arm is used as a sample problem, which is successfully optimised to use components from a fixed list while minimising cost.

Keywords Evolutionary robotics · Evolutionary computing · Standard components

---

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the NRF.

M.W. Louwrens (✉) · M.C. du Plessis · J.H. Greyling
Nelson Mandela Meteropolitan University, University Way,
Port Elizabeth 6031, South Africa
e-mail: michael.louwrens@nmmu.ac.za
URL: http://www.nmmu.ac.za

M.C. du Plessis
e-mail: mc.duplessis@nmmu.ac.za

J.H. Greyling
e-mail: jean.greyling@nmmu.ac.za

N. Pillay et al. (eds.), *Advances in Nature and Biologically Inspired Computing*,
Advances in Intelligent Systems and Computing 419,
DOI 10.1007/978-3-319-27400-3_12

1 Introduction

Evolutionary Robotics is a subset of Evolutionary Algorithms(EA) and deals with evolving morphologies and controllers for robots. Morphologies are the physical layout of the robot. The physical layout would depend on the robot type as well as the task to be performed. Each robot type can have multiple layouts where the layout of the robot includes the specific components, their placement and mounting to form a robot. The controller describes how the physical layout must work to perform the desired task.

ER has been successfully applied to evolve morphologies and controllers for several robots. The ER process has shown great promise by creating novel controllers and unique morphologies with minimal human input. However, ER has not been widely adopted by industry and the use of EA is mostly limited to merely optimising design parameters of physical robots. A possible reason for the limited adoption of ER is that it does not fit in well with the standard design process. In the traditional robotic design process the available components and overall budget are known to the designer and the design will be done within these limitations taken into account from the start. In contrast, evolving a morphology and controller using ER generates a design for a robot which must be constructed after the fact. This design does not necessarily take into account component availability or cost. The resultant robot may thus be infeasible to construct in the real world or simply be too expensive.

This research attempts to address the above mentioned shortfalls in the ER process by being a proof of concept study to demonstrate that standard off-the-shelf components can be incorporated into ER throughout the entire process. A simple robot morphology and controller is evolved which makes use of standard off the shelf components to achieve the design. This will enable the optimisation of cost along with the creation of the robot.

The robot evolved in this study is a robot arm of which the end effector must travel from a start point to an end point while avoiding an obstacle in the environment. The morphology of this arm will not be a fixed morphology. Rather the entire morphology will be evolved to suit the task and optimise obstacle avoidance, positional accuracy and cost. As this is a preliminary study this will be evolved only in simulation and the mechanical strength of the components will be ignored. Motors are thus assumed to be strong enough to lift and support the required weights.

The rest of this paper is structured as follows: Related work is discussed in Sect. 2 followed by the new proposed approach in Sect. 3. This approach will be studied using a sample problem described in Sect. 4. The implementation of this sample problem will be described in Sect. 5 after which the results will be discussed in Sect. 6 and conclusions drawn in Sect. 7.

2 Related Work

Simulation is typically used to evolve robots after which the final design is transferred to the real world. The algorithm uses a simulator to test the fitness of the robot produced. New generations are produced in the same manner that they are produced in an EA: reproduction operators. Generations of robots are created until a stopping condition is reached. The algorithm then outputs the controller and/or the body plan of the robot which then may be used in the creation of a physical robot [4].

ER can evolve novel controllers and morphologies. Lego morphologies are created by Funes [3] who shows that arbitrary morphologies can be created from basic building blocks to perform specific tasks while Lipson [8] co-evolves the robot's controller and morphology simultaneously, creating the robot's morphology and controller in a single process. Feasible robot evolution [2] investigations have shown results such that the robots produced through the ER process can be applied to real world applications. The robots used modular components which could be assembled and used to create robots designed to perform a described task.

There is lack of research which takes the physical construction of the robot into account during the ER process. An exception to this is using modular robots in the ER process. Modular robots are self-contained units which are assembled to create larger robots [1]. The modular robots are used as the base component for the chromosome in the evolutionary process [6]. The modular robot components are mirrored in the real world and the result is a robot which can be immediately assembled. Several researchers have investigated using modular robots in ER [1, 2, 6]. Despite the promising research on modular robots, modularity may limit the optimal performance of the robot [7].

The sample problem for this paper is the creation of a robot arm. Although no previous work has been done on evolving the entire structure of a robot arm, previous work has been done in optimising fixed robot arm morphologies [12, 13]. Optimising fixed arm morphologies is done by optimising link parameters such as link length or angle of twist. The goal in parameter optimisation is often to maximise the workspace of the robot arm [12, 13].

3 Proposed Approach

This paper proposes the use of standard components for robot evolution. Standard components refers to non-custom parts which are commercially available and typically inexpensive. Evolved robots can thus be simply constructed in the real world while various factors such as cost can be included in the optimisation process.

The described problem is a multi-objective optimisation problem. This is challenging as the various objectives of the algorithm must be balanced. However, in this particular instance the objectives can be categorised as either primary or secondary. Primary goals are prerequisite requirements while secondary goals are not

vital to task completion but do affect the robot. Cost would usually be a secondary goal while avoiding collisions would be a primary goal. An expensive robot can complete the task while a robot which collides with an obstacle does not.

4 Sample Problem

To show that standard components can be used in ER a sample problem of evolving a robot arm is examined. The robot arm is required to perform a single task. This task is to move from an initial position to a final position. In order to make the task and the controller more complex, an obstacle is inserted between the start and end position. The controller thus requires a path planning component which generates a safe path for the robot to traverse without colliding with the obstacle.

In papers by Rout [11], Panda [9] and Patel [10] it can be seen that typically in robot arm optimisation the structure is fixed and the parameters are continuous. This differs from this study where the structure is dynamic and the parameters are fixed by the components. Link length is not a continuous variable, it is a discrete parameter of the component selected.

The sample problem will use an obstacle starting at $(0.3, 0.3)$ and made from the two line segments $y = 0.3$ and $y = 0.3x$. This obstacle is infinitely defined in the z-axis thus the obstacle can be seen as an infinitely high wall. The path planning algorithm was required to not create new points any closer than 0.1 m to the obstacle. The target points the robot was required to reach were $(0.35, 0.1, 0.1)$ and $(0.35, 0.5, 0.2)$. The robot will have an accuracy threshold that defines how close the end-effector of the arm must be to the target point to be deemed to have successfully reached that point. This threshold value was selected to be 0.03m. This arm has primary objectives of reaching the threshold value without the arm colliding with the obstacle and a secondary objective of minimising the cost of the arm. This task is shown in Fig. 1.

A robot arm or manipulator in this scenario is a set of motors connected serially to provide a large working area. The motors are connected by links which are simple connectors which only differ in length. A discrete set of links and motors is available for the creation of the robot arm. The physical characteristics of a catalogue of commercially available servo motors was captured [5]. Mounting the components to each other is determined by three factors: a mounting type, a side to which it is mounted and an angle at which it is mounted.

5 Implementation

The goals of the sample problem were solved by splitting the problem into two free-standing evolutionary algorithms. The purpose of the first algorithm is to generate the sequence of points which would result in a collision free path. The second algorithm takes the result of the first algorithm as an input and evolves a robot arm along

with a set of motor angles which will result in the end effector reaching each point in the path. Section 5.1 describes the first algorithm while Sect. 5.2 describes the second. Various parameter choices were motivated through preliminary parameter optimisation tests. The fitness was maximised in both algorithms.

5.1 Path Finding Algorithm

5.1.1 Representation and Initialisation

The path finding EA is represented by a variable-length chromosome containing a sequence of points in three dimensional space. The chromosomes were initialised to have one to three genes upon creation. The first point in the sequence is the starting point and the final point the end point of the path. Genes were randomly initialised in a cube with one corner at the base of the robot and sides of length equal to R. R is defined as the distance between the base of the robot and the furthest goal point (either the start or end point depending on the problem).

5.1.2 Operators

A two parent crossover was used to generate a single offspring individual. Two separate random points were chosen in each parent. The offspring individual is made up of the genetic material of the first parent up to the first parent's crossover point combined with the genetic material from the second parent's crossover point up to the end of the second parent's chromosome. Tournament selection was used with a tournament size of 20 % of the population size. Two mutation operators are used, the first to introduce small amounts of local variation, while the second is used to introduce large changes. The first mutation operator adds random values from a normal distribution with a standard deviation of $\frac{R}{10}$.The second mutation operator consisted of reinitialising genes flagged for mutation. The mutation operator was applied with a probability of 0.05 with each operator having an equal chance of being used. Elitism was implemented by saving the best result of the previous generation in the new generation.

5.1.3 Fitness Function

The fitness function was created to reward individuals for creating shorter paths and punish individuals for collisions. The algorithm functions by determining if the line between two successive points intersects with the obstacle. If an intersection is found the individual is punished. The general algorithm is given in Algorithm 1

```
D = 0; P = 0;
for each successive pair of points in the chromosome: p1, p2 do
    D = D + Distance(p1,p2);
    if a collision occurs between p1 and p2 then
        P = P + 10000;
    end
end
Fitness = -(D + P);
```

Algorithm 1: Fitness of path finding algorithm

5.2 Robot Arm Evolution

5.2.1 Representation and Initialisation

Each gene in the chromosome represents a component. A component can either be a servo motor or a link, with the first component in the chromosome being a motor. The successive components in the chromosome alternate between motors and links. For each motor a sequence of N angles is stored, where N is the number of destination points found by the Path Finding Algorithm. Components are mounted based on a mounting type and variables which determine mounting information for each component is stored in the chromosome. Angles are randomly initialised within the valid travel range of the relevant servo motor, while mounting values are randomly selected from the list of valid mounting combinations. The chromosome itself was initialised to have two to ten components on creation.

5.2.2 Operators

The crossover, selection operators as well as elitism used in the path finding algorithm were used in the robot arm evolution. A restriction on the crossover was added such that the resulting child conforms to a valid representation. Two mutation operators are used in this EA. The first mutation operator potentially adds random values from a normal distribution with a standard deviation of 5° to each angle in the gene with a probability of 0.1. The second mutation operator was applied with a probability of 0.05 and consisted of reinitialising the component in genes flagged for mutation.

5.2.3 Fitness Function

The fitness function aims to create a robot arm that can successfully perform the task at the cheapest cost. The end effector of the arm must reach the threshold for all targets without colliding with the obstacle at any point while travelling through the path. The fitness of the arm then depends on the distance from the target at each

target point, the magnitude of the penetration into the obstacle and the overall cost of the arm. These separate fitnesses are weighted and added to create a single fitness for the individual. Additionally, the fitness function for the arm was designed such that it could create robots which could achieve largest distance remaining (LDR) values even lower than the threshold value. This was achieved by lowering the influence of the position of the arm once it has reached the threshold. The general algorithm is given in Algorithm 2.

Target = target list;
N = length of target list;
Component = list of components in chromosome x;
L_x = length of chromosome x;
$P_{collision} = 0$; $P_{targetdistance} = 0$; $C = 0$;
for *1 to N* **do**
  for *c = 1 to* L_x *- 1* **do**
    line = line between Component[c] and Component[c+1];
    if *line intersects with the obstacle* **then**
      $P_{collision} = P_{collision}$ + Penetration Amount;
    end
  end
end
for *n = 1 to N* **do**
  q = the end effector position when attempting to reach Target[n];
  if Distance(q,Target[n]) > Threshold **then**
    $P_{targetdistance} = P_{targetdistance}$ + Distance(q,Target[n]);
  end
  else
    $P_{targetdistance} = P_{targetdistance} + \frac{\text{Distance(q,Target[n])}}{1000}$;
  end
end
for *c = 1 to* L_x **do**
  $C = C$ + Cost(Component[c]);
end
Fitness = $-(P_{targetdistance} \times 1000000 + P_{collision} \times 1000000 + C)$;

Algorithm 2: Fitness of chromosome x

6 Evaluation

The ER process used to evaluate the sample problem was run for 30 instances. The path finding and robot arm EAs had population sizes of 100 and 50 respectively and were run for 2000 and 20,000 generations respectively. The path finding EA produced a path which added a single intermediate point to produce a collision free path as shown in Fig. 1.

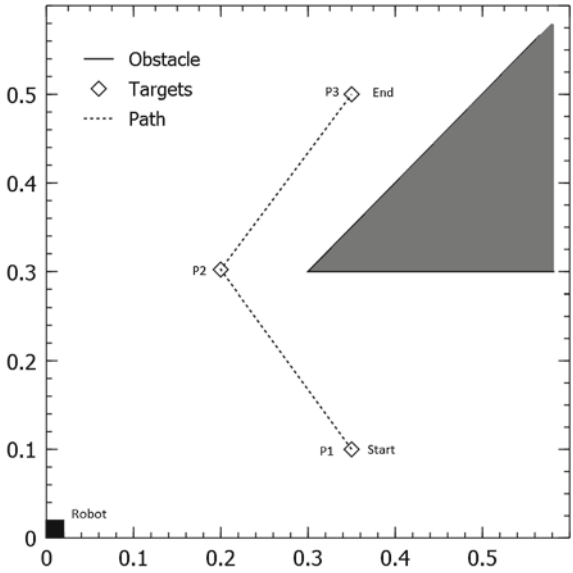

Fig. 1 Solution to the path-finding problem

Table 1 contains the output data of the 30 instances. This data is divided into multiple columns: the total magnitude of all vectors from the end effector to each target point i.e. the distance remaining summation (DRS); the largest distance remaining (LDR) which is the magnitude of the largest vector from the end effector to any target point; the sum of penetration distances(SPD) which indicates if the arm has penetrated the obstacle at any point along its path of travel; the total cost of all components in the arm; finally, the table contains the total components used to construct the arm i.e. the number of motors and links. A solution was considered successful if and only if it satisfies the primary objectives such that LDR was lower than 0.03 m and SPD was zero.

Twelve instances produced successful robots. The average cost for all thirty instances was evolved from an initial value of R1501.92 down to R601.38. The lowest cost for an arm was R332.76. This arm was the most successful by having the lowest cost while satisfying primary objectives. If minimising LDR was the goal, then the arm which had an LDR of 3.44×10^{-5}m and a cost of R482.08 would have been the most successful.

Figure 2 shows a 3D view of the best result. The LDR for this result was 0.0236 m and had a total cost of R332.76. This arm's structure had a motor at the base which rotated a link around the z-axis. The next motor was mounted at 90° to the link and would raise or lower the next link as required. The third motor was mounted inline with the previous link allowing for further height modification with a shorter link. The last motor was mounted at 90° around a different axis, rotating its link in the xy plane instead of the yz plane the previous two motors worked in. Using this layout, three configurations are shown. Each configuration represents the end effector attempting to reach P1, P2 and P3. Starting in configuration 1 the robot

Table 1 Results

DRS [mm]	LDR [mm]	SPD	Total cost	Nr. components	Notes
0.07	0.03	0	482	10	Success—most accurate
2.68	1.88	0	635	10	Success
11	10.1	0	594	13	Success
21.3	11.7	0	591	8	Success
14.4	14.3	0	603	10	Success
31.9	17	0	605	9	Success
23.3	23.2	0	566	9	Success
23.6	23.6	0	333	8	Success—least expensive
32.4	27.4	0	644	11	Success
51	28.5	0	487	12	Success
29.3	29	0	791	14	Success
36.3	29.6	0	670	10	Success
33.4	32.5	0	796	14	
34.6	34.5	0	680	8	
78.4	69.6	0	1100	10	
77.5	77.5	0	670	8	
78.2	78.1	0.00232	495	10	
102	79.6	0	700	9	
199	91	0	513	7	
114	96.3	0	563	7	
108	108	0	630	7	
122	111	0	624	8	
142	141	0	653	13	
177	146	0.000319	552	12	
149	149	0	715	9	
182	154	0	414	6	
206	186	0	449	8	
211	189	0	453	6	
253	199	0	598	6	
279	262	0	433	4	

would rotate the base and lift up the midsection of the arm to move to configuration 2. This movement shifts the end effector of the arm from P1 to P2. Moving from configuration 2 into configuration 3 requires the rotation of the base towards the obstacle and the lowering the midsection to point at P3 with the final section rotating to point in the same direction as the rest of the arm. Configuration 2 allows the arm to safely retract and extend without colliding with the obstacle. The solution shown in Fig. 2 created a workspace that could reach all the target points within the threshold value while not colliding with the obstacle at any point.

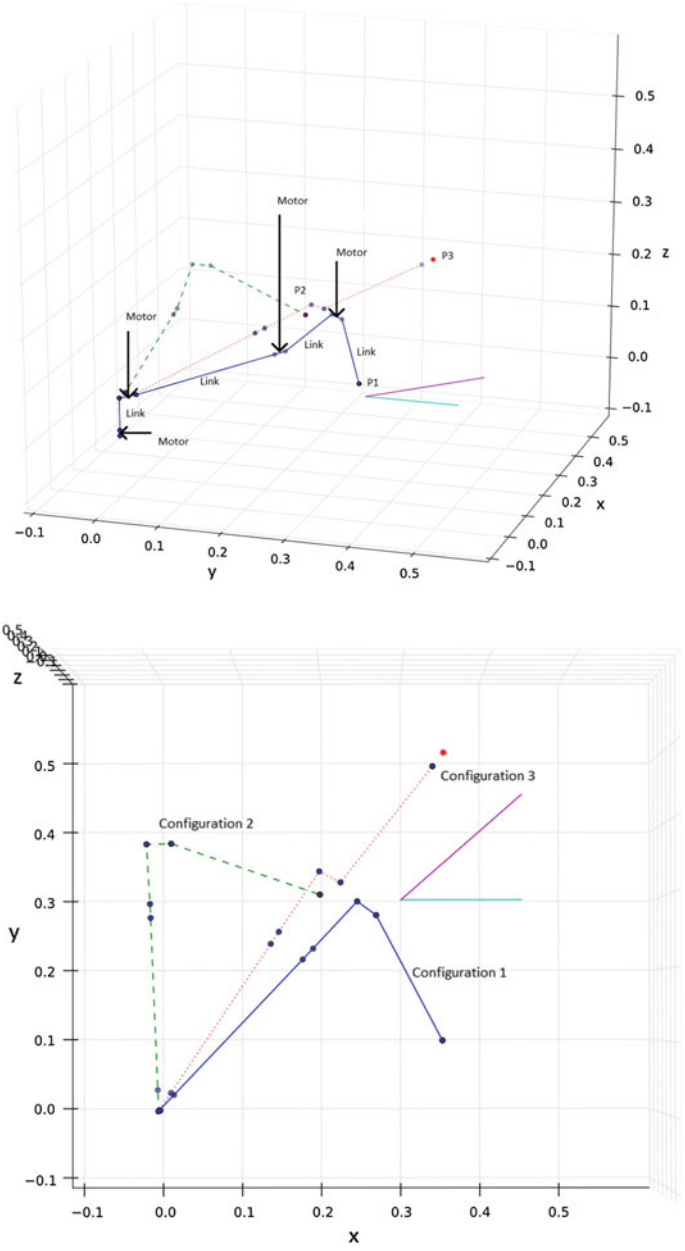

Fig. 2 Lowest cost arm *top* and orthographic views

7 Conclusion

The algorithm succeeded in producing a robot arm using the ER process and standard off-the-shelf components. 40 % of the results produced in the evaluation set were successes. The ER process succeeded in solving a multiple objective optimisation problem in that results which satisfied the primary objectives were produced while reducing cost. This is another step towards industrial application of ER.

Future work would be constructing and testing a solution in the real world. This would require the integration of torque to the fitness function. The task may or may not have a load associated with it but for the arm to be constructed in the real world, the forces acting on the arm must be calculated to ensure that the motors selected can handle the forces and possibly loads from the arm and task. This would add torque to the fitness function as a similar factor like SPD in that a robot would fail if any motor could not handle the torque acting on it.

Only rotational servos were present in the current data set, adding linear servos in future work would be an interesting addition as the results could potentially mimic a more traditional robot arm design.

References

1. Christensen, D.J., Schultz, U.P., Stoy, K.: A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots. Robot. Auton. Syst. **61**(9), 1021–1035 (2013). Sept
2. Faiña, A., Bellas, F., Orjales, F., Souto, D., Duro, R.: An evolution friendly modular architecture to produce feasible robots. Robot. Auton. Syst. **63**, 195–205 (2015). Jan
3. Funes, P., Pollack, J.: Computer evolution of buildable objects for evolutionary design by computer, pp. 1–20 (1999)
4. Harvey, I., Husbands, P., Cliff, D., Thompson, A., Jakobi, N.: Evolutionary robotics: the Sussex approach **20**, 205–224 (1997)
5. HobbyKing. Servos and Parts: http://www.hobbyking.com/hobbyking/store/__189__189__Servos_Parts.html?idCategory=189Źpc= (2015)
6. Hornby, G., Lipson, H., Pollack, J.: Generative representations for the automated design of modular physical robots. IEEE Trans. Robot. Autom. **19**(4), 703–719 (2003). Aug
7. Lipson, H.: Evolutionary robotics and open-ended design automation. Biomimetics **17**(9), 129–155 (2005)
8. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. Nature **406**(6799), 974–978 (2000)
9. Panda, S., Mishra, D., Biswal, B.: Revolute manipulator workspace optimization: a comparative study. Appl. Soft Comput. **13**(2), 899–910 (2013). Feb
10. Patel, S., Sobh, T.: Task based synthesis of serial manipulators. J. Adv. Res. (2015)
11. Rout, B., Mittal, R.: Optimal manipulator parameter tolerance selection using evolutionary optimization technique. Eng. Appl. Artif. Intell. **21**(4), 509–524 (2008). June
12. Stan, S., Balan, R., Maties, V.: Multi-objective design optimization of mini parallel robots using genetic algorithms. Ind. Electron. 2007 ISIE (1995), 2173–2178 (2007)
13. Toz, M., Kucuk, S.: Dexterous workspace optimization of an asymmetric six-degree of freedom Stewart Gough platform type manipulator. Robot. Auton. Syst. **61**(12), 1516–1528 (2013). Dec