Memetic Algorithm for Continuous Optimization

Juan Gerardo Fuentes Almeida

Centro de Investigación en Matemáticas

May 2016

Introduction

1. Human culture can be decomposed into simple units namely memes.
2. A meme is a *brick* of knowledge that can be duplicated in human brains, modified, and combined with other memes in order to generate a new meme.
3. Within a human community, some memes are simply not interesting and then will die away in a short period of time.
4. Some other memes are somewhat strong and then, similar to an infection, will propagate within the entire community.
5. The memes can also undergo slight modifications or combine with each other thus generating new memes which have stronger features and are more durable and prone to propagation.

Introduction

This interpretation of human culture inspired Moscato and Norman in late 80s, to define Memetic Algorithms (MAs) as a modification of Genetic Algorithms (GAs) employing a local search.

Basic Operations:

1. Selection of parents.
2. Combination of parents for offspring generation.
3. Local improvement of offspring.
4. Update of the population.

Memetic Algorithm

function BasicMA (**in** P: Problem, **in** par: Parameters):
Solution;
begin
    $pop \leftarrow$ Initialize(par, P);
    repeat
        $newpop_1 \leftarrow$ Cooperate(pop, par, P);
        $newpop_2 \leftarrow$ Improve($newpop_1, par, P$);
        $pop \leftarrow$ Compete ($pop, newpop_2$);
        if Converged(pop) **then**
            $pop \leftarrow$ Restart(pop, par);
        end
    until *TerminationCriterion(par)*;
    return *GetNthBest(pop, 1)*;
end

Algorithm 1: A basic memetic algorithm

The Initialize procedure is responsible for producing the initial set of $|pop|$ solutions. This could be done at random, but it is typical for MAs to attempt to use high-quality solutions as starting point, either using some constructive heuristic, or by using a local-search procedure to improve random solutions:

```
function Initialize(in par: Parameters, in P: Problem):
Bag{Solution};
begin
    pop ← Ø;
    for j ← 1 to par.popsize do
        i ← RandomSolution(P);
        i ← LocalSearch (i, par, P);
        pop ← pop ∪ {i};
    end
    return pop;
end
```

Algorithm 2: Injecting high-quality solutions in the initial population.

Memetic Algorithm

The procedures *Cooperate* and *Improve* constitute the core of the MA. *Cooperate* arises from the use of two operators for selecting solutions from the population and recombining them, it can be easily extended to use a larger collection of variation operators.

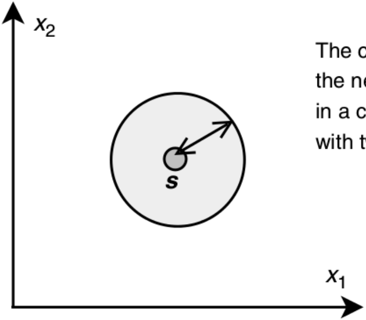
```
function Cooperate (in pop: Bag{Solution}, in par:
Parameters, in P: Problem): Bag{Solution};
begin
    lastpop ← pop;
    for j ← 1 to par.numop do
        newpop ← Ø;
        for k ← 1 to par.numapps^j do
            parents ← Select (lastpop, par.arityin^j);
            newpop ← newpop ∪ ApplyOperator (par.op^j,
            parents, P);
        end
        lastpop ← newpop;
    end
    return newpop;
end
```

Algorithm 3: The pipelined Cooperate procedure.

Memetic Algorithm

Improve embodies the application of a local search procedure to solutions in the population, considering that for continuous optimization, the neighborhood $N(s)$ of a solution s is an hypersphere with center s ans radius equal to ϵ with $\epsilon > 0$.

Hence, one have $N(s) = \{s' \in \Re^n : \| s' - s \| < \epsilon\}$ where $\| s' - s \|$ is the Euclidean norm.



The circle represents the neighborhood of s in a continous problem with two dimensions.

Memetic Algorithm

The *Compete* procedure is used to reconstruct the current population using the old population *pop* and the population of offspring *newpop*2. There exist two main possibilities for this purpose: the *plus strategy* and the *comma strategy*:

1. Plus Strategy: The current population is constructed taken the best popsize configurations from $pop \cup newpop$.

2. Comma Strategy: The best popsize configurations are taken just from newpop. In this case, it is required to have $|newpop| > popsize$, so as to put some selective pressure on the process.

Memetic Algorithm

I order to apply a restart procedure, it must be decided whether the population has degraded or has not, using some measure of information diversity in the population. A very typical strategy is to keep a fraction of the current population, and generate new (random or heuristic) solutions to complete the rest.

function Restart (**in** *pop*: Bag{Solution}, **in** *par*: Parameters,
in *P*: Problem): Bag{Solution};
begin
    $newpop \leftarrow \emptyset$;
    for $j \leftarrow 1$ **to** $par.preserved$ **do**
        $i \leftarrow$ GetNthBest(pop, j);
        $newpop \leftarrow \{i\}$;
    end
    for $j \leftarrow par.preserved + 1$ **to** $par.popsize$ **do**
        $i \leftarrow$ RandomSolution(P);
        $i \leftarrow$ LocalSearch (i, par, P);
        $newpop \leftarrow \{i\}$;
    end
    return $newpop$;
end

Algorithm 4: The restart procedure.

As for the *TerminationCriterion* function, one of the following criteria could be used:

1. Checking a limit on the total number of iterations.
2. Reaching a maximum number of iterations without improvement.
3. Having performed a certain number of population restarts.
4. Reaching a certain target fitness.

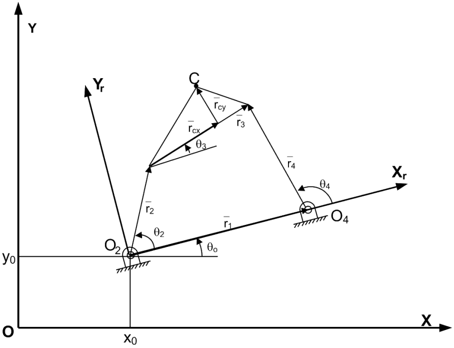
Problem Description

Four-bar Mechanism

The objective is that the terminal C reches as much as possible a set of target points $\{C_d^i\}$.

The engine torque is in θ_2, while $r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, x_0, y_0$ and θ_0 are design parameters to be optimized.

Given all these parameters, it is possible to calculate the coordinates of the terminal C.



Problem Description

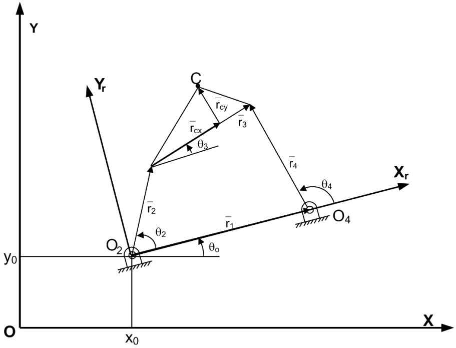
Four-bar Mechanism

With respect to the coordinate system in x_0, y_0 and θ_0:

$$\begin{aligned} \widehat{C}_r &= \widehat{r}_2 + \widehat{r}_{cx} + \widehat{r}_{cy} \\ C_{xr} &= r_2 \cos\theta_2 + r_{cx}\cos\theta_3 - r_{cy}\sin\theta_3 \\ C_{yr} &= r_2 \sin\theta_2 + r_{cx}\sin\theta_3 + r_{cy}\cos\theta_3 \end{aligned}$$

Transforming to the coordinate system **O**xy:

$$\begin{bmatrix} C_x \\ C_y \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} C_{xr} \\ C_{yr} \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$



Problem Description

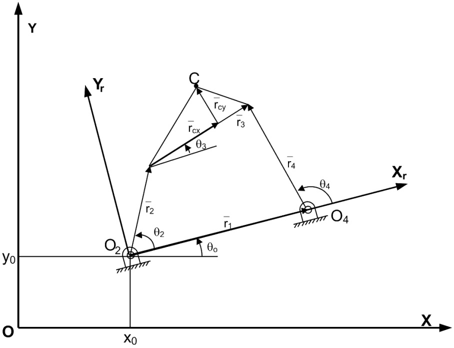
Four-bar Mechanism

Close loop Equations for the Mechanism:

$$\widehat{r}_1 + \widehat{r}_4 = \widehat{r}_2 + \widehat{r}_3$$
$$r_2 \cos\theta_2 + r_3 \cos\theta_3 = r_1 + r_4 \cos\theta_4$$
$$r_2 \sin\theta_2 + r_3 \sin\theta_3 = r_4 \sin\theta_4$$

squaring both equations:

$$r_4^2 \cos^2\theta_4 = (r_2 \cos\theta_2 + r_3 \cos\theta_3 - r_1)^2$$
$$r_4^2 \sin^2\theta_4 = (r_2 \sin\theta_2 + r_3 \sin\theta_3)^2$$



Problem Description

By summing the equations and applying trigonometric identities we obtain:

$$r_4^2 = r_1^2 + r_2^2 + r_3^2 + 2r_2r_3\cos(\theta_2 - \theta_3) - 2r_1r_3\cos\theta_3 - 2r_1r_2\cos\theta_2$$

By rearranging and redefining terms we formulate the expresion called *Freudenstein's Equation*:

$$k_1\cos\theta_3 + k_2\theta_2 + k_3 = \cos(\theta_2 - \theta_3)$$

Where:

$$k_1 = \frac{r_1}{r_2}; \quad k_2 = \frac{r_1}{r_3}; \quad k_3 = \frac{r_4^2 - r_1^2 - r_2^2 - r_3^2}{2r_2r_3}$$

Problem Description

Let be $\varphi = tan(\frac{\theta_3}{2})$, then

$$\sin\theta_3 = \frac{2\varphi}{1+\varphi^2}; \quad \cos\theta_3 = \frac{1-\varphi^2}{1+\varphi^2}$$

Substituing in the Freudenstein Equation and rearranging terms:

$$k_1(\frac{1-\varphi^2}{1+\varphi^2}) + k_2\cos\theta_2 + k_3 = (\frac{1-\varphi^2}{1+\varphi^2})\cos\theta_2 + \frac{2\varphi}{1+\varphi^2}\sin\theta_2$$

$$\varphi^2[k_3 + (k_2+1)\cos\theta_2 - k_1] + \varphi(-2\cos\theta_2) + [k_1 + (k_2-1)\cos\theta_2 + k_3] = 0$$

$$\Rightarrow \varphi = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Problem Description

Where:

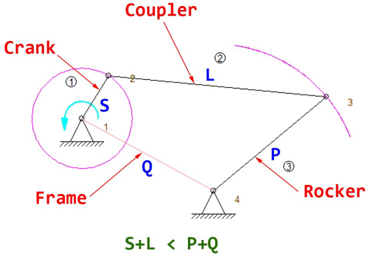
$$\begin{aligned} a &= k_3 + (k_2+1)\cos\theta_2 - k_1 \\ b &= -2\cos\theta_2 \\ c &= k_1 + (k_2-1)\cos\theta_2 + k_3 \end{aligned}$$

Then, θ_3 can be computed as $\theta_3 = 2tan^{-1}\varphi$

If $\varphi \notin \Re$, the mechanism cannot be constructed with the given configurations.

Mechanism Restrictions.

1. Input angles θ_2^i should be arranged such that they are consecutive in a 2π radians circumference.
2. All link magnitudes shall be positive: $r_1, r_2, r_3, r_4, r_{cx}, r_{cy} > 0$.
3. Grashof Condition imposed to form a *Crank-Rocker* configuration, that is $r_1 + r_2 < r_3 + r_4$, with $r_2 < r_3, r_4 < r_1$, and the shortest link is connected to the base.



In order to use this definition of the problem when the optimization algorithm is implemented, the constraints 1,2 & 3 are retained when values are assigned to design variables, and also they are inserted into the goal function as penalty functions as follows:

$$min\{[(C^i_{xd}(X) - C^i_x(X))^2 + (C^i_{yd}(X) - C^i_y(X))^2] + M_1 h_1(X) + M_2 h_2(X)\}$$

where:
$X = [r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, \theta_0, \theta_2^1, \theta_2^2, ..., \theta_2^N]$

$h_1(X) \in \{0, 1\} \rightarrow$ Grashof Condition true/false.
$h_2(X) \in \{0, 1\} \rightarrow$ Sequence Condition for θ_2 true/false.

and M_1 and M_2 are constants of a very high value that penalize the goal function when the associated constraint fails.

Implementation

Implemented Operators:

Recombination. It can be defined as a process in which a set S_{par} of n configurations (parents) is manipulated to create a set $S_{desc} \subseteq sol_P(x)$ of m new configurations (descendants).

The creation of these descendants involves the identification and combination of features extracted from the parents. In this implementation we use the *Dynastically Optimal Recombination* scheme, in which every recombination is explored in order to find the best configuration, the result will be at least equal to the best parent.

We are also considering a partial recombination, that is, we are only considering blocks of information when recombining individuals, namely, the sets $\{r_1, r_2, r_3, r_4\}$, $\{r_{cx}, r_{cy}\}$, $\{x_0, y_0, \theta_0\}$ and $\{\theta_2^1, \theta_2^2, ..., \theta_2^N\}$

Implementation

Mutation. It creates a single offspring x_i' from each parent x_i, $\forall i \in \{1, 2, ..., popsize\}$ by

$$x_i'(j) = x_i(j) + \epsilon N_j(0, 1)$$

where j is a random integer $\in \{1, 2, ..., popsize\}$, and $N_j(0, 1)$ is a random value generated from a standard normal distribution.

Results

The following table shows the best results obtained from 100 executions of the algorithm.

Target points: $C_d^i = \{(20,20),(20,25),(20,30),(20,35),(20,40),(20,45)\}$
Limits: $r_1, r_2, r_3, r_4, r_{cx}, r_{cy}, x_0, y_0 \in [0,60]$; $\theta_0, \theta_2^1, \theta_2^2, \theta_2^3, \theta_2^4, \theta_2^5, \theta_2^6 \in [0, 2\pi]$
Parameters: $popsize = 100$, $MaxIte = 1000$, $\epsilon_r = 10$, $\epsilon_\theta = 0{,}1$

r1	r2	r3	r4	rcx	rcy	x0	y0	theta0	Fmin
55.996129	11.860317	26.684302	56.960179	30.735369	9.883953	3.142643	60	3.639357	0.0078
59.875522	10.727668	24.577835	58.406998	26.280596	12.157331	5.141042	56.566359	3.560392	0.019058
21.780566	16.606229	23.301379	35.007264	31.545113	8.443519	1.877289	17.435899	2.131829	0.056205
48.925425	9.471383	14.60938	60	14.385883	3.037092	10.834641	23.031599	2.919015	0.057158
59.357222	15.341739	31.193179	53.203131	31.443674	10.591183	10.075585	60	3.631323	0.068617
54.545358	12.828405	30.386007	60	22.220154	3.507703	22.268048	54.857281	5.834672	0.123284
58.653971	12.605048	23.248806	59.017517	25.818171	2.040868	17.393864	6.96745	2.719125	0.127332
48.205104	11.643272	22.048389	60	21.951581	7.189605	10.931092	12.262803	2.706078	0.135521
50.17406	9.547689	14.303315	48.069709	11.681369	12.891571	27.696188	19.179331	0.099839	0.136301
18.452372	12.747353	11.731494	26.301019	19.475191	8.622404	0.262667	47.633093	3.815419	0.141773
52.293613	5.36955	15.977059	57.319538	18.300585	23.233571	44.530481	20.244287	6.298891	0.151417
47.218007	7.353142	17.068789	46.920148	14.20552	23.388206	39.890086	18.257959	6.40881	0.165234
43.488833	11.53675	23.21043	53.100963	23.014422	6.763177	11.933853	10.399746	2.636528	0.167028
59.991858	11.549794	23.951936	57.012862	24.562784	8.267317	11.411802	56.637731	3.527898	0.173476
59.857459	14.703776	18.057195	59.966778	17.321549	6.453634	13.04751	45.753718	3.383988	0.1749
39.824146	11.749458	32.012771	60	27.747639	13.609629	5.615179	5.188327	2.492471	0.175453
32.777872	10.658131	21.95618	53.432359	15.250522	9.850886	8.52869	23.64741	2.712092	0.202074
20.110768	7.897732	32.785112	35.031658	20.60465	9.357022	35.098083	20.165608	0.763898	0.205023
29.810541	8.889684	50.570899	41.56882	34.653848	2.929273	42.69272	7.769873	1.333422	0.210038
46.149813	11.615341	20.813982	39.843859	12.552755	17.613575	4.851433	49.275176	3.519957	0.210232

References

1 *J.A. Cabrera & A. Simon, M. Prado* (2002). **Optimal synthesis of mechanisms with genetic algorithms.** Journal of Mechanism and Machine Theory 37 (2002) 1165-1177.

2 *El-Ghazali Talbi* (2009). **Metaheuristics: form Design to Implementation.** John Wiley & Sons, Inc.

3 *Pablo Moscato & Carlos Cotta* (2003). **A Gentle Introduction to Memetic Algorithms.** Handbook of Metaheuristics, Kluwer Academic Publishers, pp. 105-144

4 *Kumar Chellapilla* (1998). **Combining Mutation Operators in Evolutionary Programming.** IEEE Transactions on Evolutionary Computation, Vol. 2, No. 3.

5 *Ferrante Neri & Carlos Cotta* (2012). **Memetic algorithms and memetic computing optimization: A literature review.** Swarm and Evolutionary Computation 2, 114.