Constrained Molecular Dynamics as a Search and Optimization Tool

Riccardo Poli[1] and Christopher R. Stephens[2]

[1] Department of Computer Science, University of Essex, UK
[2] Instituto de Ciencias Nucleares, UNAM, México

Abstract. In this paper we consider a new class of search and optimization algorithms inspired by molecular dynamics simulations in physics.

1 Introduction

Search and optimization algorithms take inspiration from many different areas of science. For instance, evolutionary algorithms generically take their inspiration from biological systems [1]; simulated annealing from the physics of cooling [2]; Hopfield neural networks from the physics of spin glasses [3,4]; swarm algorithms from social interactions [5,6]. In this paper we consider a class of algorithms that take their inspiration from physics, in the sense that they use of a group of interacting particles to perform the search, and will consider how intuition from there can help in understanding how they work. Systems of this type have been widely studied and simulated in the field of *molecular dynamics*.[1]

Our system is somehow similar to (and partly inspired by) particle swarm optimization (PSO) [5]. Like our system, PSOs use groups of interacting particles. In PSOs such particles *fly over* the fitness landscape, recording the best places seen so far by each particle and by the swarm as a whole. These points are then used to generate pseudo-random forces (produced by springs) which attract the particles towards such points. Although other types of interactions have been recently introduced [9], generally no other information on the fitness landscape is used by PSOs.

Our system differs from a PSO in many ways. Firstly, the motion of our particles is *constrained to be on the fitness landscape*, that is our particles *slide on* the fitness landscape rather than fly over it. Secondly, our simulation is physically realistic, in the sense that a variety of forces may act on our particles such as gravity, friction, centripedal acceleration, in addition to coupling forces such as those generated by springs connecting the particles. As will be shown later some of these forces depend on the topological characteristics of the landscape, such as its gradient and curvatures, in the neighbourhood of each particle. Thirdly, our method does not require the presence of explicit intelligence in the particles,

[1] The molecular dynamics method [7], introduced over 40 years ago to study the interactions of hard spheres [8], has since been widely used in physics to understand the behaviour of liquids, proteins, DNA, enzymatic reactions, etc.

unlike PSOs where this is necessary for observing the motion of the particles and deciding when and how to change the position of the attractors. Fourthly, the method does not rely on the use of pseudo-random forces to perform the search (although these are not excluded).

Given the close relation with molecular dynamics and the constrained nature of the motion of our particles, we have decided to term the class of algorithms considered here *Constrained Molecular Dynamics* (CMD).

The paper is organized as follows. In the next section we describe the basic principles behind CMD. In Section 3 we illustrate the effects of different types of forces on CMD using simple examples. We look at some implementation details in Section 4. In Section 5 we study the behaviour of the algorithm on a small set of benchmark problems. We give our conclusions in Section 6.

2 Constrained Molecular Dynamics

We restrict attention here to continuous search spaces. The search space we denote by $V \subset \mathcal{R}^N$, where $\mathcal{R}^N$ is an N-dimensional Euclidean space and equip it with coordinates $\{x_1, \dots, x_N\}$. We also consider a fitness function $f : \mathcal{R}^N \to \mathcal{R}^+$. We can now embed V in $\mathcal{R}^{N+1}$, defining the fitness landscape via $y = f(\mathbf{x})$, where y is a height function above V and we use bold face to denote vectors in V. In terms of the embedding coordinates, $\mathbf{r} = \{x_1, \dots, x_N, x_{N+1}\}$, where $y = x_{N+1}$, the fitness function surface takes the form $g(x_1, \dots, x_N, x_{N+1}) = y - f(x_1, \dots, x_N) = 0$. Using a notation that is standard in physics we will denote derivatives of f by $f_{,i_1 \dots i_m} = \partial^m f / \partial x_{i_1} \dots \partial x_{i_m}$, e.g. $f_{,i} = \partial f / \partial x_i$, where we use Roman indices for components of vectors associated with V. For indices associated with V we will also use the Einstein summation convention that repeated indices on different objects are considered to be summed over, e.g. $x_i x_i = \sum_i x_i x_i$.

We now consider n particles of mass m^α, $\alpha \in \{1, \dots, n\}$, where α denotes the particle under consideration[2], moving on the surface $y = f(\mathbf{x})$, with (embedding) coordinates and velocities $\mathbf{x}^\alpha$ and $\mathbf{v}^\alpha$ respectively. In this case the motion of each particle is constrained via an equation $g(\mathbf{x}^\alpha) = 0$. Thus, we have $(N+1)n$ coordinates and N constraints, which leads to Nn independent degrees of freedom.

The kinetic energy, T^α, of a particle α is

$$T^\alpha = \frac{m^\alpha}{2} |\mathbf{v}^\alpha|^2 = \frac{m^\alpha}{2} \left(\dot{x}_i^\alpha \dot{x}_i^\alpha + \dot{x}_{N+1}^\alpha \dot{x}_{N+1}^\alpha \right) \tag{1}$$

where $\dot{x}_i^\alpha = dx_i^\alpha / dt$ and $\dot{x}_{N+1}^\alpha = f_{,i} \dot{x}_i^\alpha$. (Generally l dots above the symbol will represent l time derivatives.) Then

$$T^\alpha = \frac{m^\alpha}{2} g_{ij}^\alpha \dot{x}_i^\alpha \dot{x}_j^\alpha \tag{2}$$

[2] Greek indices will be used to specify the particle of interest.

with $g^\alpha_{ij} = \delta_{ij} + f^\alpha_{,i} f^\alpha_{,j}$ being interpretable as the metric tensor for the space on which the particles are moving. The equations of motion for the particle α are

$$\frac{d}{dt}\left(\frac{\partial T^\alpha}{\partial \dot{x}^\alpha_i}\right) - \frac{\partial T^\alpha}{\partial x^\alpha_i} = \mathbf{F}^\alpha \cdot \frac{\partial \mathbf{r}^\alpha}{\partial x^\alpha_i} \tag{3}$$

where $\mathbf{F}^\alpha$ is the force on the particle. Explicitly

$$\ddot{x}^\alpha_i + \Gamma^\alpha_{ijk} \dot{x}^\alpha_j \dot{x}^\alpha_k = h^\alpha_{ij} \frac{F^\alpha_k}{m^\alpha} r^\alpha_{k,j} \tag{4}$$

where (h^α_{ij}) is the inverse of (g^α_{ij}) and

$$\Gamma^\alpha_{ijk} = \frac{f^\alpha_{,i} f^\alpha_{,jk}}{(1 + f^\alpha_{,i} f^\alpha_{,i})} \tag{5}$$

The quantity Γ^α_{ijk} is determined solely by geometrical properties of the fitness landscape and gives rise to the "generalized" force that arises due to the constrained motion on the surface. Equation (3) gives a complete description of the dynamics of particle α. The particle trajectories are solutions of (3). The questions now are: can a set of particles like these perform a search, what type of search do these particles carry out and how good are the particles at finding optimal points in the landscape?

3 "Forces for Courses"

If one wishes to use the above physical system for search and optimization it behooves one to think about what would be useful properties to have in order to perform such tasks well. To a large extent this is associated with what type of forces are introduced into the particle dynamics, as well as such obvious characteristics as the number of particles.[3] In the above we have not specified the forces. There are, of course, a huge variety of possibilities. We may fruitfully think of several broad classes however: i) no forces; ii) forces due to particle-particle interactions; iii) forces due to interactions with an external field and iv) friction/viscosity type forces. In the limit when there are no forces then equation (4) becomes

$$\ddot{x}^\alpha_i + \Gamma^\alpha_{ijk} \dot{x}^\alpha_j \dot{x}^\alpha_k = 0 \tag{6}$$

Here the effective generalized force on the particle arises purely due to its constrained motion. This generalized force depends on the geometry of the landscape via Γ^α_{ijk}, which from (5) can be seen to be zero when $f_{,i} = 0$ or $f_{,jk} = 0$. A simple example of this is a particle constrained to move on a half circle (the lower half) of radius R. In V, this is one dimensional motion, hence there is only one degree of freedom. The constraint is $y = -(R^2 - x^2)^{\frac{1}{2}}$. In this case Γ^α_{ijk} only has one component, $\Gamma^\alpha_{111} = x/(R^2 - x^2)$. The equation of motion is

$$\ddot{x} + \left(\frac{x}{R^2 - x^2}\right) \dot{x}^2 = 0 \tag{7}$$

[3] In the rest of the paper we will assume $m^\alpha = 1$ for all particles.

This can most easily be solved in a polar coordinate system (r, θ), where we take θ to be the deviation from the direction $-\mathbf{e}_y$ and $\mathbf{e}_y$ is a unit vector in the y direction. In this case (7) becomes

$$\ddot{\theta} = 0 \tag{8}$$

whose solution is $\theta(t) = a + bt$. Thus, in this coordinate system, naturally given the geometry, the particle moves freely as there is no generalized force in the tangential direction. The particle position in terms of the landscape height is $y(t) = -R\cos(a+bt)$. In terms of "optimization", if we are seeking the minimum of the function then obviously if the particle starts with zero velocity we have the trivial situation where the particle does not move. However, moving with angular frequency ω clockwise and starting at $y = 0$ then $y(t) = -R\cos(\pi/2 - \omega t)$. The particle finds the optimum when $\cos(\pi/2 - \omega t) = 1$, i.e. $t = \pi/2\omega$. So in order to find the optimum the particle needs a non-zero initial velocity.

Considering further this simple one-dimensional example we can now introduce the idea of an external force field. A canonical example of that would be gravity, of which the simplest case is that of a constant gravitational field. In the case of particle motion on a surface it is natural to take the gravitational acceleration, g, to be in the y direction. In this case $F_i^\alpha = -g\delta^\alpha_{i(N+1)}$, i.e. the force is "downwards". Once again, taking the case $y = -(R^2 - x^2)^{\frac{1}{2}}$, one finds

$$\ddot{x} + \left(\frac{x}{R^2 - x^2}\right)\dot{x}^2 = \frac{gx}{R}(R^2 - x^2)^{\frac{1}{2}} \tag{9}$$

Passing to a polar coordinate system we have

$$\ddot{\theta} = -g\sin\theta \tag{10}$$

Intuition can be simply gleaned here by considering the case $\theta \ll 1$ so that $\sin\theta \approx \theta$. In this case $\theta(t) = a\cos(g^{1/2}t) + b\sin(g^{1/2}t)$. Starting the particle at $\theta(0) = a$ $(a > 0)$ with inital angular speed zero the particle finds the optimum $\theta = 0$ at $t = \pi/2g^{1/2}$, which is independent of the initial starting point. Thus, adding a constant external force that pulls the particles in the required direction, i.e. to smaller values of the height function, implies that the particle can find the global optimum irrespective of the particle's initial position. Thus, in terms of optimization the advantage of gravity is that it provides a bias for the search to go in the right direction. One might be tempted to think of it as providing a hill climbing type behaviour. This would be wrong however, as (some) *local optima can be avoided.* This can be simply understood in the one-dimensional case by realizing that dropped from a given height with zero velocity the particle will be able to surmount any local maxima that are lower than its starting point as the particle's accumulated kinetic energy is sufficient to take it over the barrier.

In order to consider particle-particle interactions we must go beyond the one particle case, the simplest being that of two particles. An interesting and illustrative example of interparticle interactions would be to connect the particles by attractive spring type forces, where generically the force on particle α would be $\mathbf{F}^\alpha = -\sum_\beta k d_{\alpha\beta}$, where k is a spring constant which in principle could be different for different particles. The sum over β is a sum over those particles β connected to the particle α. This could be all the particles or just nearest neighbors or a random subset, to name but a few. Finally, $d_{\alpha\beta}$ is the "signed" distance between α and β, where $d_{\alpha\beta} = -d_{\beta\alpha}$. This could be the Euclidean distance in the embedding space, the distance in V or the distance as measured by a geodesic curve between the two particles (i.e. the shortest distance on the landscape). In the case of our simple example of a particle constrained to move on a half circle the two equations of motion for the two particles are

$$\ddot{x}^\alpha + \left(\frac{x^\alpha}{R^2 - (x^\alpha)^2}\right)(\dot{x}^\alpha)^2 = -k d_{\alpha\beta} \tag{11}$$

where $\alpha = 1, 2$. In the case where the spring force is associated with the distance between the particles as measured along the curve then passing to polar coordinates (r_1, θ_1) and (r_2, θ_2) and introducing the center of mass and relative coordinates $\Theta = (\theta_1 + \theta_2)/2$ and $\theta_r = (\theta_1 - \theta_2)/2$ one finds

$$\ddot{\Theta} = 0 \qquad\qquad \ddot{\theta}_r = -k\theta_r \tag{12}$$

In this case the center of mass moves with uniform angular speed $\Theta = a + bt$ while the relative angle between the two particles is given by $\theta_r = c\cos(k^{1/2}t) + d\sin(k^{1/2}t)$. A simple example of how such interparticle forces can help in the search process can be to consider the case where both particles start on either side of the optimum at $\theta = 0$, either at rest or with velocities that take them away from the optimum. In this case the attractive force of the spring pulls them in the direction of the optimum. In the explicit example where the particles start with initial positions and velocities $\theta_1 = a$, $\theta_2 = -a$, $\dot{\theta}_1 = \dot{\theta}_2 = 0$ then the particles encounter the optimum at $t = \pi/2k^{1/2}$.

Finally, in discussing the individual forces in the context of simple examples we may consider the case of friction. Taking the particle on the half circle the equation of motion is

$$\ddot{\theta} = -\eta\dot{\theta} \tag{13}$$

which has solution $\theta(t) = (b\eta - a + a\exp(-t))/\eta$, where $\dot{\theta}(0) = -a$ and $\theta(0) = b$, i.e. we start the particle from the right of the optimum and travelling toward it. In the limit of large t the particle will have reached the optimum if $\eta b < a$. Thus, the stronger the friction force the greater the initial velocity in the direction of the optimum must be in order to reach it. It may be though naively then that friction is a bad idea. However, say for example, $b < 0$ then the presence of friction prevents the particle from moving further away from the optimum than it would otherwise do. In this way, in the presence of other forces, friction can have an important role to play in "relaxing" the particle into a good position once an interesting region has been found in the landscape. Increasing the friction then has the character of reducing the explorative component in the search and is somewhat analogous to reducing the temperature in the case of simulated annealing.

Now, in the above we are considering a simple example of a unimodal function in one dimension. We have in mind using the particles for search and optimization on non-trivial multi-dimensional landscapes. So what can we deduce from the above in the more general context? Firstly, consider free particle motion. This is somewhat analogous to random search. There is no "selection" in the sense that there is no systematic tendency to seek fitter (lower) points in the landscape. In this case the particle's trajectory is completely and solely governed by the geometry of the landscape. However, there is a tendency to spend less time in regions of the landscape of high curvature and more time in regions of low curvature. Intuitively this is because the "tighter the bend the quicker the particle has to travel to keep on the track". In this sense the particle motion can be thought of as a potential diagnostic for the size of the basin of attraction of an optimum.

Other type of forces can be included in this framework. For example, particles could be charged as has been proposed for PSOs [9]. In our simulations we have used elasticity (partially unelastic bounces) also to guarantee that the particles would not leave the search area defined by the user.

4 Discretization of the Algorithm

Although Equation (3) gives a complete description of the dynamics of each particle, its explicit solution is generally very hard if not impossible for a generic landscape and a generic set of forces. So, often numerical integration is the only way to determine the trajectory of each particle and, therefore, to run a CMD algorithm. In our implementation we have used the traditional first-order forward difference approximation for derivatives, e.g. we have approximated a continuous velocity $v(t)$ as

$$v(t) = \frac{x(t+\Delta) - x(t)}{\Delta},$$

where Δ is the integration time step.

So, once the initial velocity and position of each particle is given, we update each component of velocity and position of each particle, time step after time step, by using the following recursion:

$$x(t+1) = x(t) + \Delta \cdot v(t) \quad (14)$$

$$v(t+1) = v(t) + \Delta \cdot a(t) \quad (15)$$

where $a(t)$ is the corresponding component of the acceleration. This is calculated by appropriately adding all the forces (gravity, friction, etc.) acting on the particle. Some such forces depend on the first or second order partial derivatives of the fitness function. These can either be directly computed by differentiation of the fitness functions or be approximated numerically. For simplicity in our experiments we calculated derivatives using central differences. For example, we used the approximation

$$f_{,i} \approx \frac{f(x_1, \ldots, x_i + \Delta, \ldots x_n) - f(x_1, \ldots, x_i - \Delta, \ldots x_n)}{2\Delta}.$$

Naturally, these extra evaluations of the fitness function need to be considered when calculating the computation load of the algorithm.

5 Results

We tested the algorithm on three standard benchmark problems, the De Jong's functions F1 and F2, and the Rastrigin's function, of increasing dimensionality N and with a varying number of masses n. The method we propose is new, and still needs to be understood in depth. So, the purpose of these tests was not to try and beat other well established algorithms, but rather to understand more about what kind of forces are beneficial for what kind of landscapes and why.

The function F1 has the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} x_i^2.$$

That is this function represents a symmetric (hyper-)paraboloid, with no local optima and a global optimum in $\mathbf{x} = (0, \dots, 0)$ where $f(\mathbf{x}) = 0$.

The function F2 (also known as Rosenbrook's function) has the following form:[4]

$$f(\mathbf{x}) = \sum_{i=1}^{N-1} ((x_{i+1} - x_i^2)^2 + 0.01(1 - x_i)^2).$$

This function has no local optima and a global optimum in $\mathbf{x} = (1, \dots, 1)$ where $f(\mathbf{x}) = 0$. The function is much harder than F1 since the optimum is effectively at the end of a long, very narrow valley.

Rastrigin's function has the following form:

$$f(\mathbf{x}) = 10N + \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i))$$

This function has many local optima and one global optimum in $\mathbf{x} = (0, \dots, 0)$ where $f(\mathbf{x}) = 0$.

In the simulations reported here, we initialized all the particles in random positions within the hypercube $[-5.12, +5.12]^N$, all with zero velocities.[5]

For each of the three fitness functions described above we tested a number of different configurations:

- different numbers of particles ($n = 1$, $n = 2$ and $n = 10$),
- different dimensionality of the search space ($N = 1$, $N = 2$ and $N = 3$),
- different configurations for the springs (absence, particles connected so as to form a ring, particles fully connected: s=no, s=ring and s=full, respectively, in Tables 1–3),

[4] For simulation convenience we rescaled the original function by dividing it by 100.

[5] Note, however, that providing the particles with non-zero initial velocities can have benefits since it energises the system beyond what's provided by the gravitational and elastic potential energies.

– absence or presence of gravity (g=no and g=yes, respectively, in Table 1–3),
– absence or presence of friction (f=no and f=yes, respectively, in Tables 1–3).

For each setting we performed 30 independent experiments. Each experiment involved the integration of the system of equations in Equation (3) for 5000 time steps with time step $\Delta = 0.01$. Spring stiffness was 0.03, the friction coefficient was 1, gravity acceleration was 0.1, the elasticity coefficient for bounces against the boundaries -5.12 or +5.12 was 0.8. In each run we measured the average and standard deviation of the best fitness value found during the run, as well as the average and standard deviation of the distance between the point where the best value was achieved and the global optimum.

Tables 1–3 report the results of the simulations. Each entry includes four numbers. The first (in italics) represents the average of the best fitness seen in each of the 30 independent runs. The second (in a smaller font and in italics) represent the standard deviation of the best fitness. The third value represents the average distance from the global optimum, while the fourth (in a smaller font) represents the standard deviation for such a distance.

For F1 (see Table 1) we note that, as expected, increasing the number of particles performing the search improves performance. This is true for all our experimental results (i.e. also for Tables 2 and 3). We can also see that, alone, the presence of gravity is sufficient to guarantee near perfect results. This had to be expected since a particle with no initial velocity is bound to pass through the origin in this fitness function. Springs appear not to be too beneficial for the search, particularly in the case of only two particles where the system tends to oscillate in useless directions, unless gravity brings the system (and its oscillations) in the right area. Friction generally helps the search settle in the global optimum.

In Table 2 we report the results obtained with the much harder De Jong's function 2. Generally the comments above apply to this function too, although, due to the long narrow valley leading to the global optimum, the benefits of friction are not as clear in this case. Also, for this function we observe a significant variance in the results for the case $N = 3$. This is due to the fact that, depending on the initial conditions, runs either succeed in finding (and then following until the simulation ends) the bottom of the valley or they oscillate between the "walls" of the valley failing to ever get good fitness values. With enough particles the effect disappears, because there always appear to be some which are well placed to find the valley. For this problem the presence of springs benefits the system when only two particles are present (particularly if gravity is also present). This is because the interaction between the particles help align the trajectories of the particles in the direction of the valley.

The results for the Rastrigin function (Table 3) appear to be the worst of the crop. This is really due to the exceptional multimodality of this function (e.g. for $N = 3$ the function presents around 1000 local optima in the interval of interest). In all cases gravity appears to help bring the system towards the global optimum more than anything else, although the presence of fully connected springs seems to have potential (because the particles can then pull each other out of local

Table 1. Results on De Jong fitness function 1. The cases where no motion could happen are marked as N/A.

	N=1			N=2			N=3		
Setup	n=1	n=2	n=10	n=1	n=2	n=10	n=1	n=2	n=10
s=no	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
g=yes	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
f=no	0.0017	0.0010	0.0001	0.0038	0.0023	0.0005	0.0047	0.0031	0.0005
	0.0012	0.0009	0.0002	0.0029	0.0023	0.0005	0.0032	0.0025	0.0005
s=no	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
g=yes	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
f=yes	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0008	0.0002	0.0000
	0.0000	0.0000	0.0000	0.0002	0.0001	0.0000	0.0014	0.0008	0.0000
s=ring	N/A	*0.6463*	*0.0000*	N/A	*0.8151*	*0.0014*	N/A	*0.8539*	*0.0074*
g=no		*1.5581*	*0.0000*		*3.0105*	*0.0022*		*4.1464*	*0.0090*
f=no		0.3225	0.0001		0.3449	0.0294		0.3821	0.0736
		0.7364	0.0002		0.8343	0.0234		0.8414	0.0441
s=ring	N/A	*0.8501*	*0.0000*	N/A	*1.0403*	*0.0039*	N/A	*1.0241*	*0.0076*
g=no		*1.8996*	*0.0000*		*3.6969*	*0.0099*		*4.2864*	*0.0117*
f=yes		0.2360	0.0001		0.4165	0.0427		0.4726	0.0724
		0.6492	0.0002		0.9311	0.0460		0.8948	0.0485
s=ring	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0158*	*0.0002*	*0.0000*	*0.0346*	*0.0012*
g=yes	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0220*	*0.0003*	*0.0000*	*0.0489*	*0.0013*
f=no	0.0018	0.0013	0.0001	0.0038	0.0951	0.0099	0.0047	0.1513	0.0303
	0.0013	0.0016	0.0002	0.0029	0.0820	0.0080	0.0032	0.1083	0.0172
s=ring	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0010*	*0.0000*	*0.0000*	*0.0028*	*0.0000*
g=yes	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0045*	*0.0000*	*0.0000*	*0.0067*	*0.0000*
f=yes	0.0000	0.0000	0.0000	0.0001	0.0120	0.0003	0.0008	0.0309	0.0017
	0.0000	0.0000	0.0000	0.0002	0.0286	0.0004	0.0014	0.0433	0.0018
s=full	N/A	*0.8058*	*0.0000*	N/A	*1.9934*	*0.0064*	N/A	*1.2609*	*0.0073*
g=no		*1.8267*	*0.0000*		*5.0823*	*0.0280*		*4.5858*	*0.0132*
f=no		0.3661	0.0002		0.6871	0.0392		0.5760	0.0687
		0.8196	0.0002		1.2334	0.0698		0.9639	0.0512
s=full	N/A	*1.1956*	*0.0000*	N/A	*2.8333*	*0.0505*	N/A	*1.5146*	*0.0123*
g=no		*1.9363*	*0.0000*		*5.6838*	*0.2465*		*4.7949*	*0.0172*
f=yes		0.4302	0.0002		0.8495	0.0876		0.6795	0.0892
		0.6727	0.0002		1.2733	0.2070		1.0261	0.0662
s=full	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0052*	*0.0000*	*0.0000*	*0.0174*	*0.0002*
g=yes	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0090*	*0.0001*	*0.0000*	*0.0210*	*0.0003*
f=no	0.0018	0.0008	0.0001	0.0038	0.0522	0.0049	0.0047	0.1065	0.0132
	0.0013	0.0008	0.0002	0.0029	0.0496	0.0039	0.0032	0.0777	0.0078
s=full	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0004*	*0.0000*	*0.0000*	*0.0009*	*0.0000*
g=yes	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0021*	*0.0000*	*0.0000*	*0.0032*	*0.0000*
f=yes	0.0000	0.0000	0.0000	0.0001	0.0054	0.0002	0.0008	0.0132	0.0008
	0.0000	0.0000	0.0000	0.0002	0.0194	0.0002	0.0014	0.0266	0.0005

Table 2. Results on De Jong fitness function 2. This function is not defined for $N = 1$.

	N=1			N=2			N=3		
Setup	n=1	n=2	n=10	n=1	n=2	n=10	n=1	n=2	n=10
s=no	N/A	N/A	N/A	*0.0083*	*0.0057*	*0.0008*	*98.6604*	*24.5817*	*0.0540*
g=yes				*0.0063*	*0.0057*	*0.0012*	*168.2136*	*70.6019*	*0.1733*
f=no				1.5725	1.2344	0.5024	3.2142	2.3850	1.8800
				0.9038	0.8570	0.4947	1.8425	1.1993	0.8406
s=no	N/A	N/A	N/A	*0.0085*	*0.0062*	*0.0011*	*98.6783*	*24.5529*	*0.0281*
g=yes				*0.0063*	*0.0061*	*0.0017*	*168.2172*	*70.6264*	*0.0406*
f=yes				1.8870	1.4017	0.4768	3.2300	2.4520	1.7980
				1.1955	1.0719	0.3980	1.8333	1.1514	0.7935
s=ring	N/A	N/A	N/A	N/A	*0.0116*	*0.0119*	N/A	*2.0799*	*0.1160*
g=no					*0.0101*	*0.0234*		*9.4327*	*0.2155*
f=no					1.4533	1.3980		2.0027	1.9016
					0.8073	1.2000		1.0316	0.9785
s=ring	N/A	N/A	N/A	N/A	*0.0093*	*0.0102*	N/A	*2.0842*	*0.0884*
g=no					*0.0079*	*0.0186*		*9.4467*	*0.1197*
f=yes					1.4369	1.4064		1.9913	1.9155
					0.8969	1.1284		1.0333	0.9686
s=ring	N/A	N/A	N/A	*0.0083*	*0.0091*	*0.0026*	*98.6604*	*0.2926*	*0.0285*
g=yes				*0.0063*	*0.0091*	*0.0039*	*168.2136*	*0.7559*	*0.0215*
f=no				1.5725	1.3385	0.7636	3.2142	1.8632	1.0944
				0.9038	0.8018	0.6339	1.8425	0.6457	0.5917
s=ring	N/A	N/A	N/A	*0.0085*	*0.0096*	*0.0056*	*98.6783*	*0.2768*	*0.0567*
g=yes				*0.0063*	*0.0102*	*0.0140*	*168.2172*	*0.7184*	*0.0711*
f=yes				1.8870	1.2516	0.7584	3.2300	1.8642	1.7421
				1.1955	0.7705	0.6502	1.8333	0.6534	0.9999
s=full	N/A	N/A	N/A	N/A	*1.4072*	*0.0179*	N/A	*2.2154*	*0.1057*
g=no					*6.6479*	*0.0279*		*10.6180*	*0.3041*
f=no					1.5476	1.5874		2.0587	1.5411
					1.0641	1.3333		1.0573	0.6969
s=full	N/A	N/A	N/A	N/A	*1.4429*	*0.0146*	N/A	*2.1228*	*0.1086*
g=no					*6.7059*	*0.0233*		*10.6202*	*0.2967*
f=yes					1.5831	1.7296		2.0383	1.5132
					1.0120	1.1779		1.0003	0.7283
s=full	N/A	N/A	N/A	*0.0083*	*0.0072*	*0.0020*	*98.6604*	*0.2107*	*0.0345*
g=yes				*0.0063*	*0.0085*	*0.0023*	*168.2136*	*0.4660*	*0.0510*
f=no				1.5725	1.1053	0.7714	3.2142	1.7438	1.5821
				0.9038	0.6276	0.6215	1.8425	0.7965	0.6131
s=full	N/A	N/A	N/A	*0.0085*	*0.0068*	*0.0016*	*98.6783*	*0.2513*	*0.0346*
g=yes				*0.0063*	*0.0073*	*0.0023*	*168.2172*	*0.5929*	*0.0524*
f=yes				1.8870	1.1092	0.5870	3.2300	1.9117	1.5445
				1.1955	0.6805	0.4682	1.8333	0.8000	0.6408

Table 3. Results on Rastrigin's function.

	N=1			N=2			N=3		
Setup	n=1	n=2	n=10	n=1	n=2	n=10	n=1	n=2	n=10
s=no g=yes f=no	*0.0332* *0.0469* 0.3331 0.4685	*0.0266* *0.0440* 0.2669 0.4398	*0.0066* *0.0248* 0.0668 0.2481	*1.0396* *0.5970* 2.8790 0.9066	*0.7361* *0.5427* 2.2777 1.0634	*0.3201* *0.2693* 1.4607 0.7793	*1.9737* *0.9226* 4.0112 1.1571	*1.5414* *0.7397* 3.5864 0.9498	*0.8132* *0.4943* 2.6426 0.8851
s=no g=yes f=yes	*0.3980* *0.0000* 1.9899 0.0000	*0.2919* *0.1518* 1.5919 0.6079	*0.0929* *0.1283* 0.6633 0.6957	*1.1536* *0.6872* 3.1834 1.0348	*0.7825* *0.5866* 2.4916 1.1448	*0.3307* *0.2901* 1.5797 0.7683	*2.0360* *1.0148* 4.3004 1.1970	*1.5813* *0.8024* 3.7682 1.0576	*0.7773* *0.4447* 2.4908 0.8670
s=ring g=no f=no	N/A	*0.3368* *0.6520* 0.6223 1.0560	*0.0000* *0.0000* 0.0003 0.0003	N/A	*0.9024* *0.6795* 2.2698 1.1799	*0.5437* *0.4631* 1.8193 0.9733	N/A	*2.1433* *0.9289* 3.5220 1.1853	*1.2487* *0.5519* 3.0507 0.9802
s=ring g=no f=yes	N/A	*0.4286* *0.6698* 0.8941 1.1796	*0.0000* *0.0000* 0.0003 0.0004	N/A	*1.1336* *0.8059* 2.5529 1.3239	*0.4475* *0.3490* 1.6278 0.7376	N/A	*2.0803* *1.0903* 3.3908 1.2126	*1.1415* *0.5226* 2.9809 0.9300
s=ring g=yes f=no	*0.0332* *0.0469* 0.3331 0.4685	*0.1029* *0.2430* 0.4322 0.9147	*0.0000* *0.0000* 0.0005 0.0006	*1.0396* *0.5970* 2.8790 0.9066	*1.1870* *0.7936* 2.7213 1.2532	*0.3327* *0.2492* 1.5775 0.8007	*1.9737* *0.9226* 4.0112 1.1571	*1.9231* *0.9096* 3.5778 1.2404	*0.8747* *0.4806* 2.6596 0.8376
s=ring g=yes f=yes	*0.3980* *0.0000* 1.9899 0.0000	*0.1928* *0.3036* 0.8012 1.0773	*0.0000* *0.0000* 0.0003 0.0004	*1.1536* *0.6872* 3.1834 1.0348	*0.9684* *0.6065* 2.4871 1.1120	*0.2835* *0.2392* 1.4193 0.8035	*2.0360* *1.0148* 4.3004 1.1970	*1.8926* *0.8526* 3.6725 1.1526	*0.8497* *0.5028* 2.6898 0.9011
s=full g=no f=no	N/A	*0.3913* *0.6669* 0.6997 1.1208	*0.0000* *0.0000* 0.0002 0.0002	N/A	*1.1838* *0.9084* 2.6754 1.3083	*0.4292* *0.2601* 1.6599 0.8089	N/A	*2.1011* *1.0658* 3.7625 1.1425	*1.2331* *0.5655* 2.9868 0.9181
s=full g=no f=yes	N/A	*0.4286* *0.6698* 0.8943 1.1796	*0.0000* *0.0000* 0.0003 0.0004	N/A	*1.1066* *0.8272* 2.5687 1.3165	*0.4671* *0.3094* 1.8347 0.8264	N/A	*1.8753* *1.0624* 3.6730 1.1208	*1.1929* *0.5441* 2.8970 0.8626
s=full g=yes f=no	*0.0332* *0.0469* 0.3331 0.4685	*0.0579* *0.1306* 0.3031 0.6409	*0.0000* *0.0000* 0.0005 0.0007	*1.0396* *0.5970* 2.8790 0.9066	*0.8969* *0.7295* 2.3166 1.2982	*0.3358* *0.2609* 1.5941 0.7446	*1.9737* *0.9226* 4.0112 1.1571	*1.5931* *0.7710* 3.5718 1.0327	*0.8681* *0.4602* 2.7054 0.8320
s=full g=yes f=yes	*0.3980* *0.0000* 1.9899 0.0000	*0.1228* *0.1698* 0.6973 0.8572	*0.0166* *0.0371* 0.1661 0.3707	*1.1536* *0.6872* 3.1834 1.0348	*0.8517* *0.7193* 2.4117 1.2834	*0.3373* *0.2761* 1.6053 0.7694	*2.0360* *1.0148* 4.3004 1.1970	*1.4464* *0.7426* 3.2468 1.1666	*0.7913* *0.4474* 2.5512 0.8517

optima, at least in some cases). Judging from the rapid improvements produced by increasing the number of particles from 2 to 10 in all cases, we suspect that further significant improvements could be obtained by using a larger number of particles. Future research will need to clarify this.

6 Conclusions

In this paper we have presented a new search and optimization method inspired by nature: the Constrained Molecular Dynamics method. This method uses the physics of masses and forces to guide the exploration of fitness landscapes. In this paper we have started exploring this idea, using three forces: gravity, interaction via springs, and friction. Gravity provides the ability to seek minima. Springs provide exploration. Friction slows down the search and focuses it.

In the paper we have described experimental results for the De Jong's functions 1 and 2 and for the Rastrigin's function. The results appear to be very encouraging and make us believe that a lot more can come from this method in future research.

Acknowledgements. CRS acknowledges support from DGAPA project ES100201 and Conacyt project 41022-E. RP would like to thank the members of the NEC (Natural and Evolutionary Computation) group for helpful comments and discussion.

References

1. J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA, 1975.
2. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.
3. J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.
4. J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.
5. James Kennedy and Russell C. Eberhart. *Swarm Intelligence.* Morgan Kaufmann Publishers, San Francisco, California, 2001.
6. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence : from natural to artificial systems.* Oxford University Press, New York, 1999.
7. D. C. Rapaport. *The Art of Molecular Dynamics Simulation.* Cambridge University Press, 1997.
8. B. J. Alder and T. E. Wainwright. Phase transition for a hard sphere system. *Journal of Chemical Physics*, 27:1208–1209, 1957.
9. T. M. Blackwell and P. J. Bentley. Dynamic search with charged swarms. In W. B. Langdon *et al.*, editor, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 19–26, New York, 9-13 July 2002. Morgan Kaufmann Publishers.