09/04/06

Problem: The problem is not that PSO cannot be applied. ~~it is~~

The problem resides in properly defining the space in which PSO is to be ~~defined~~ applied. And determining a proper relation between the behavior space and the physical space with PSO in relation to that.

private void probehavior()
{

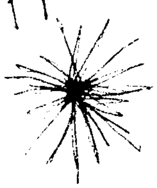
if (!stopped)
  selectRule();

  r = Math.random();

~~r = Math~~ r =>
  setV(φ_i * Math.random() * getV());
  setVx(φ_L * r * getVx());
  setVy(φ_L * r * getVy());

pL = maxVisibility = 0
~~maxVisibility~~
pG = maxUtility = 0

for (Planet p : planetList)
{



~~maxVisibility~~ =
if (maxVisibility < p.getVisibility())
  pL = maxVisibility = p.getVisibility(); // p_L = the position associated with the max visibility (x, y)

~~if (maxVisib < p.get~~
if (maxUtility < p.getUtility())
  pG = maxUtility = p.getUtility(); // p_G = the position associated with the max utility (x, y)

} endfor
rL = Math.random();
rG = Math.random();
~~setV(getV + φ_i * Math.random~~() rL *

setVx(getVx + φ_L * rL * (pLx − x(t)) + φ_G * rG * (pGx − x(t))
setVy(getVy + φ_L * rL * (pLy − y(t)) + φ_G * rG * (pGy − y(t))
~~setVx(getVx() + φ_L * rL *~~ setV(getV() + φ_L * rL * (pL − current position) + φ_G * rG * (pG − current position)

case 0:
  if (this.getVx() > Vmaxx = 650)
    setVx(Vmax X);
  if (this.getVx() < 0)
    setVx(0);
  if (this.getVy() > Vmaxy = 450)
    setVy(Vmax y);

  if (this.getVy() < 0)
    setVy(0);

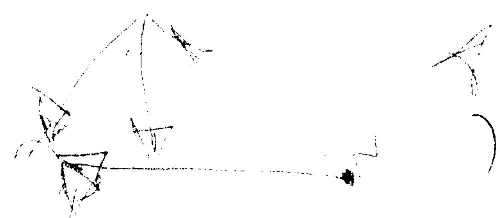
I don't like this configuration

$x(t+1) = \varphi_i\,\mathrm{rand}_i()\,v(t)$
$+ \varphi_L\,\mathrm{rand}_L()(p_L - x(t))$
$+ \varphi_G\,\mathrm{rand}_G()(p_G - x(t))$

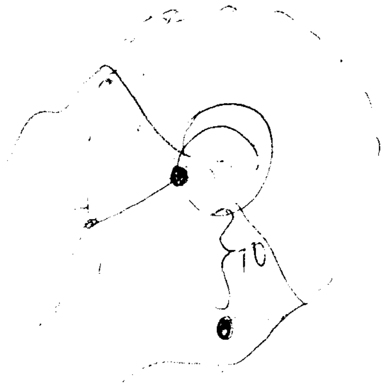
$p_L = p(V_{max}(t))$
$p_G = p(U_{max}(t))$

case 1:

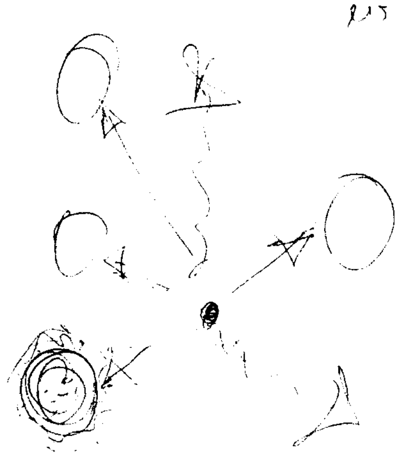
```
                 envDims
if (getVx() > VmaxX =640)
  do
      setVx( ~~this getX()~~
       640 - this.getPosition().getX());
  ~~if getVx~~ while ( ~~getVx()~~ >640)

                envDimsX - envDims X
if ( getVx() < ~~VmaxX~~ =0)

   do
     setVx(                              );

   while (getVx() < 0)

if (getVy() > VmaxY = 480)
   do
     setVy( 480 - this.getPosition().getX()
   while ( getVy > 480)

if ( getVy() < 0)
   do
     setVy(                              );
```

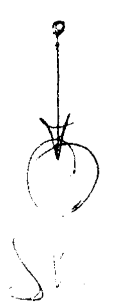


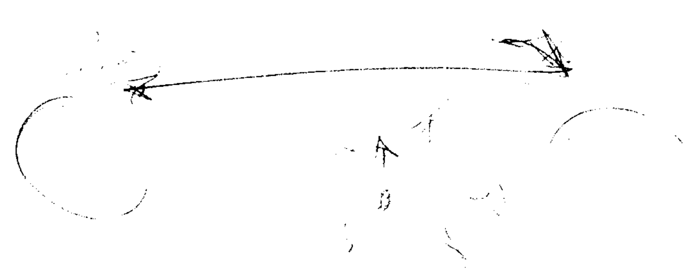
case 2:

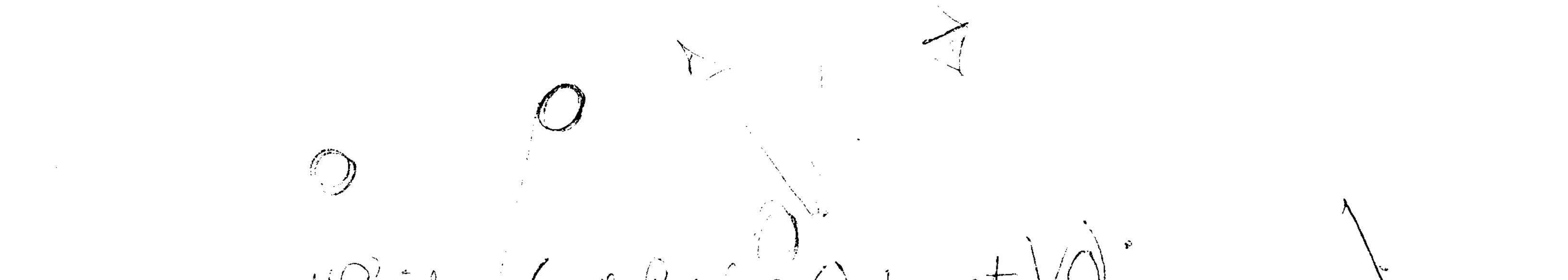



```
if (getVx > Vmax x = 640)
    do setVx (getVx - 640)
    while (getVx > 640)

if (getVx < 0 )
    do setVx(getVx + this getPosition().getX());
    while ( getVx < 0 )

if (getVy() > Vmaxy = 480)
    do setVy (getVy - 480)
    while (getVy > 480)

if ( getVy() < 0 )
    do setVy (getVy + this.getPosition().getY());
    while ( getVy < 0 )
```

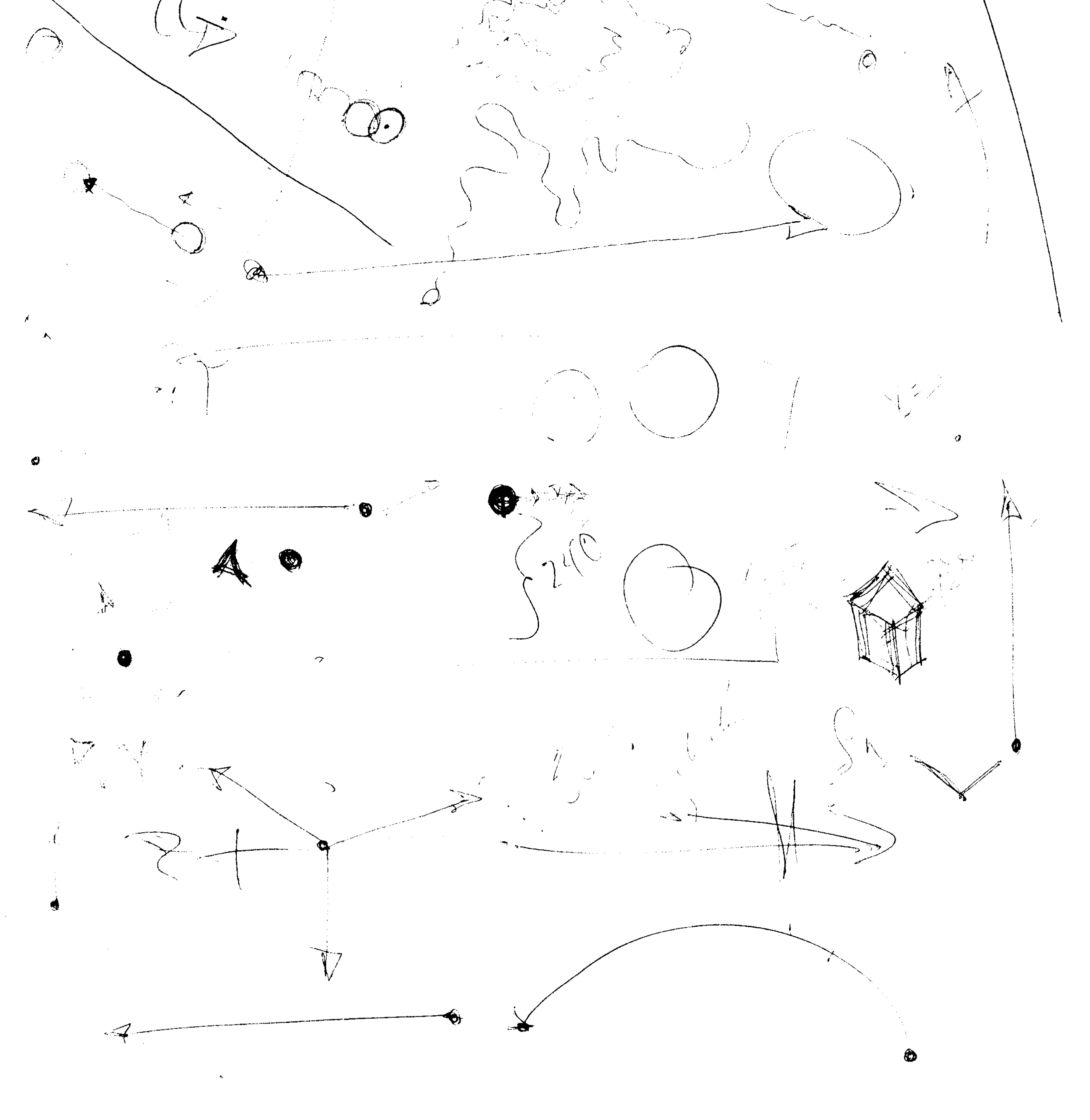








settPosition(gettPosition()+ getV());



checkTargets();

{

if (~~within~~ target(s) within visible range)

```
if ((status = B) && (t is active) {
        blah, blah, blah!
        status = H;
}
else if ((status = T) && (t is active)) {
        status = T;
}
else if ((status = B) && (t is not active) {
    continue status = B // Don't need to set BeenHereDoneThat
}                          // ~~Don't lock~~ to ~~out the pheromone~~
else if ((status = T) && (t is not active) {
    continue status = B // set BeenHereDoneThat on the
}  // associated pheromone
```

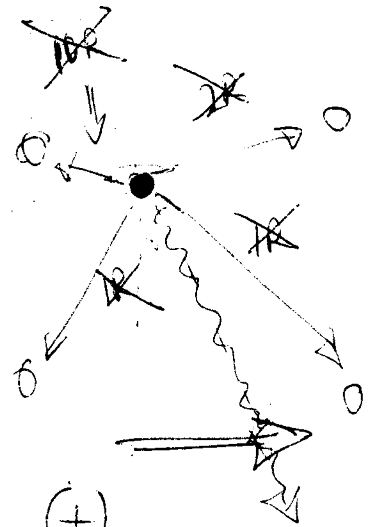
Boss running

set()	checkTargets()	harvest()
The behavior is committed in set().	The behavior is set in checkTargets().	The behavior is set in harvest().

EFFAHPHAT

09 | 04 | 06

#1



if $(v > v_{max})$

~~$v = v_{max}$~~

$v = v_{max}$;

else if $(v < -v_{max})$

$v = -v_{max}$;

if $(v_x > v_{x,max})$

$v_x = v_{x,max}$

else if $(v_x < -v_{x,max})$

$v_x = -v_{x,max}$

if $(v_y > v_{y,max})$

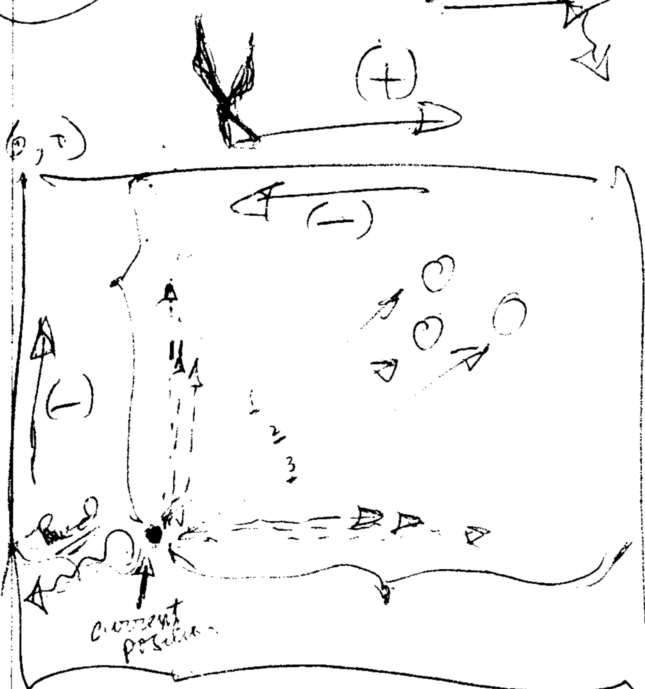
$v_y = v_{y,max}$

else if $(v_y < -v_{y,max})$

$v_y = -v_{y,max}$

EFFAHPHAT

if (+)



$(-x_{max} \le v_{max} \le x_{max})$

What is $[-v_{max}, v_{max}]$?

Should it be defined by the boundaries of the environment (640 x 480 pixels) or the targets (i.e. $-x_{max} \Rightarrow$ minimum x amongst the x-vectors and the same for y.)

$x_{max} = (640) - (\text{current } x\text{-position})$ (640, 480)

$-x_{max} = -(\text{current } x\text{-position})$

② if $(v_x > 640)$ $v_x = (640) - (\text{current } x\text{-position})$

if $(v_x < 0)$ $v_x = -(\text{current } x\text{-position})$

if $(v_y > 480)$ $v_y = (480) - (\text{current } y\text{-position})$

if $(v_y < 0)$ $v_y = -(\text{current } y\text{-position})$

$$v_{xy}(t+1) = \varphi_R \, Rand_R() \, v_{xy}(t) + \sum \varphi_i \, rand_i() \, v_i(M_i(t))$$

if $(x < 0)$ v_x = bounce speed;

if $(x >$ width $= 640)$ $v_x = -$ bounce speed;

if $(y < 0)$ v_y = bounce speed;

if $(y >$ height $= 480)$ $v_y = -$ bounce speed;

$$\mu = \frac{\eta(\omega - \tau)}{R} = \frac{\text{visibility}(\text{~~food~~ weight ratio} - \text{intensity})}{\text{redundancy}}$$

$$\mu = \frac{(\omega - \tau)}{dR} = \frac{(\text{food weight ratio} - \text{intensity})}{(\text{distance} * \text{redundancy})}$$

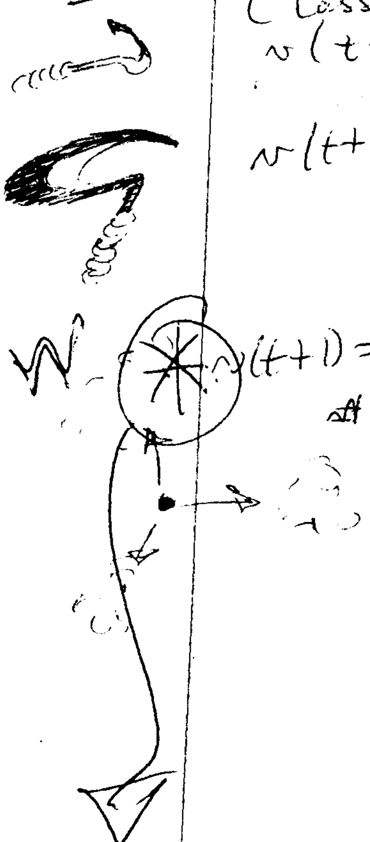
Classic PSO

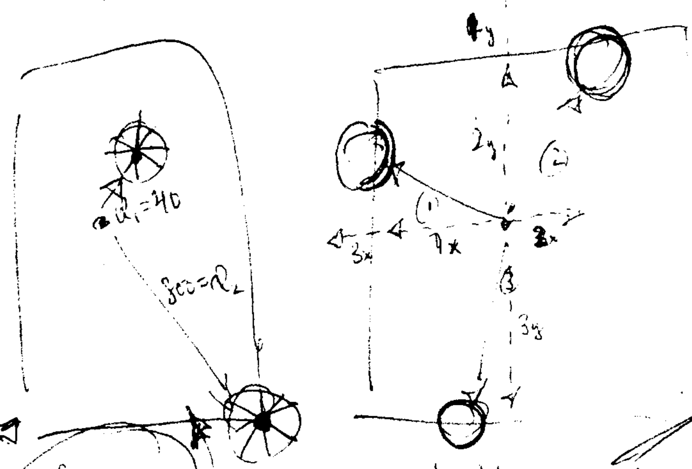
$$v(t+1) = \varphi_k\, rand_k()\, v(t) + \varphi_L\, rand_L()(p_L - x(t)) + \varphi_G\, rand_G()(p_G - x(t))$$

$$v(t+1) = \varphi_k\, Rand_k()\, v(t) + \sum_{i=0}^{\text{phast size}=I} \varphi_i \mu_i$$

$$= \varphi_k\, Rand_k()\, v(t) + \sum \varphi_L\, rand_L(p_L(\mu_L(t)) - x(t)) + \sum \varphi_G\, rand_G()(p_G(\mu_G(t)) - x(t))$$

$$v(t+1) = \varphi_k\, Rand_k()\, v(t) + \sum \varphi_i\, rand_i(p_i(\mu_i(t)) - x(t))$$





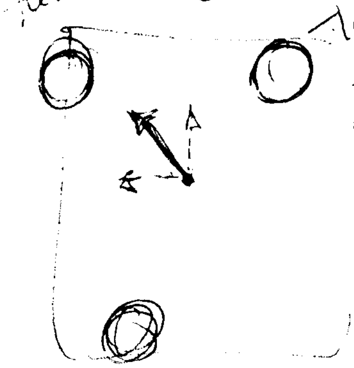
$$v_x(t+1) = \varphi_k\, Rand_k()\, v_x(t) + \sum \varphi_i\, rand()_i(p_{i,x}(\mu_i(t)) - x(t))$$

$$\frac{1.00}{40} = 0.025$$

$$v_y(t+1) = \varphi_k\, Rand_k()\, v_y(t) + \sum \varphi_i\, rand()_i(p_{i,y}(\mu_i(t)) - y(t))$$

$$\frac{1.000}{530} = 0.00125$$

resultant



~~However,~~ closer targets are made to be farther away based on redundancy

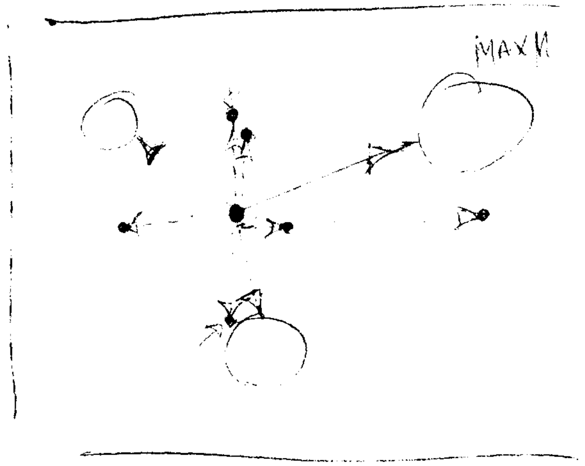
$$v(t+1) = \sqrt{v_x^2(t+1) + v_y^2(t+1)}$$

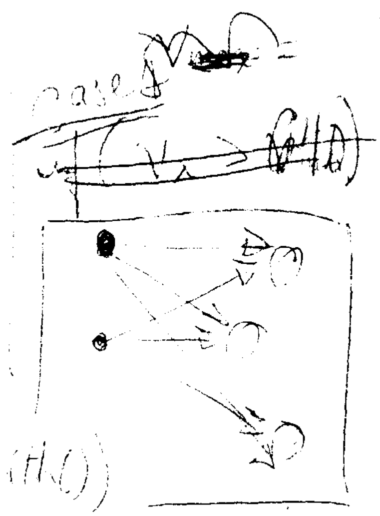
$$\theta(t+1) = \arctan(v_y(t+1)/v_x(t+1))$$

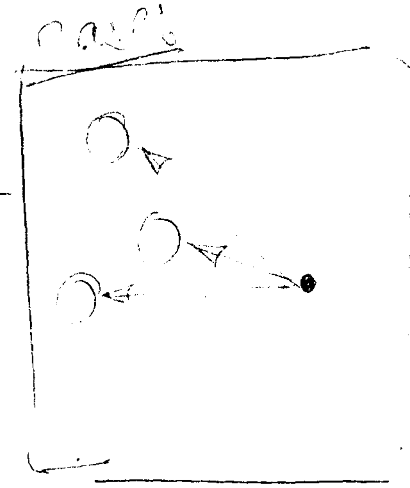
$$\Rightarrow \vec{v}(t+1)$$

$$x(t+1) = x(t) + \vec{v}(t+1)$$

Determine Vmax game entity







if (V_x > (640 = [illegible]Dims.getWidth())

~~setVx([illegible].getTarget().getVx( )~~

setVx(maximumVelocity().getTarget().getPosition(). ~~getVx~~ = this.getPosition().getVx());

if (V_x < 0)

setVx(

if (V_y > (480 = [illegible]Dims.getHeight())

setVy(

Share indirect visibility and direct visibility.

Important?

(A) [illegible]

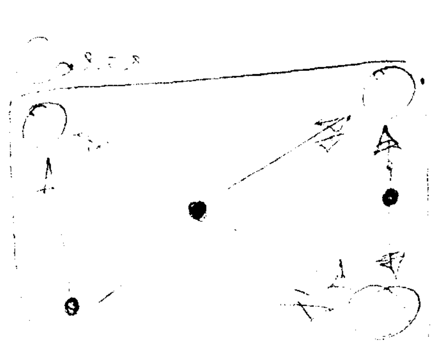
(1) [illegible]

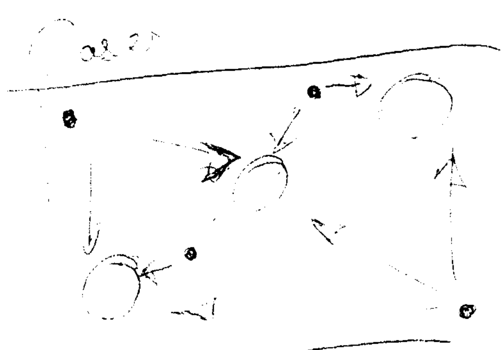
(2) [illegible]

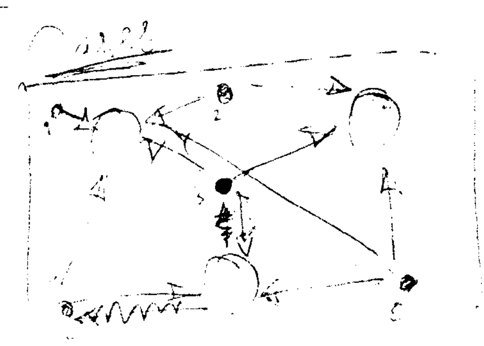
(3) (A) ≠ (B) ≠ (C)

(4) (A) ≠ (B) (C) ≠

(5) (A) ≠ [illegible] ≠ (C)

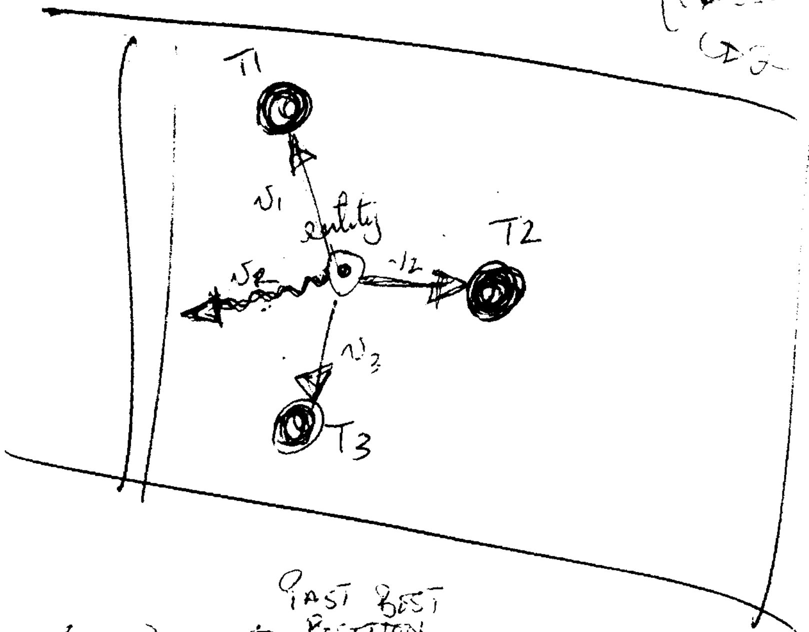






$[0.0 \ldots 1.0] * (5-2)^3 + 2$

(2) 2 to 5 DOMAINS



$\mu = \frac{\omega - t}{dk}$

x, y

PAST BEST POSITION

$v(t+1) = \varphi_R \text{rand}()\, v(t) + \varphi_2$

| 09 | 07 | 06 |

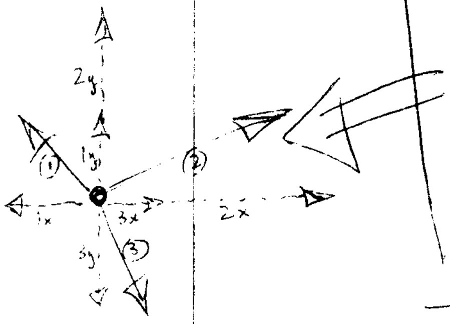
#2

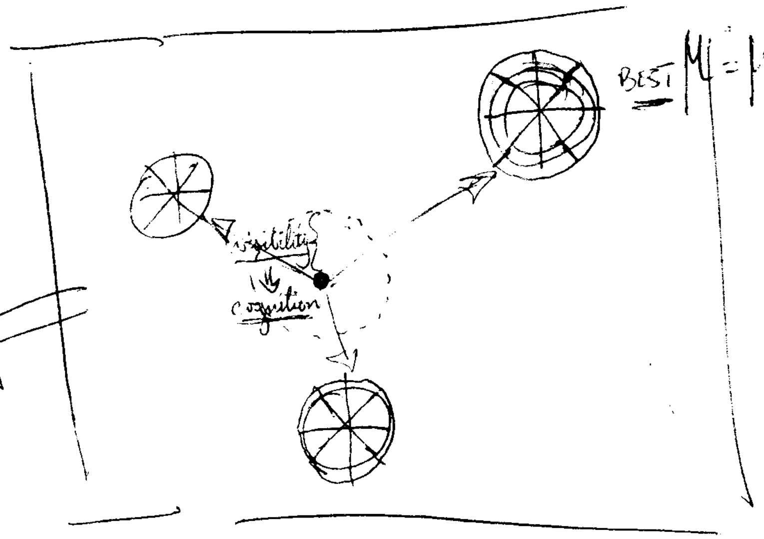
$v(t+1) = \varphi_R \text{Rand}()\, v(t) + \sum \varphi_i \text{rand}_i (v_i(\mu_i(t)))$

public void psoBehavior() {
Rand = Math.random(); // from negative to positive values.
vx = φ_R Rand vx;
vy = φ_R Rand vy;
for (Pheromone p : pheromoneList) {

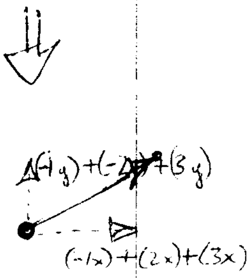
~~vx+=~~

vx+=
~~position~~ =
vy+=





BEST $M_i = M_{gBEST}(t) = M_g(t)$



$(p_G - x_{ij}(t))$

$$v_{ij}(t+1) = \varphi_R * rand_R() * v_{ij}(t) + \varphi_L * rand_L() * (p_L - x_{ij}(t)) + \varphi_G * rand_G() *$$

past best — cognition ⇒ max visibility — global ⇒ max

$$M_{ij}(t+1) = \varphi_R * rand_R() * M_{ij}(t) + \varphi_L\, rand_L() * (M_{ij}(t) - x_{ij}(t)) + \varphi_G\, rand_G() *$$

max visibility — $(\omega_{ij}(t) - x_{ij}(t))$ ⇒ max pool weight ratio

? $rand_R()$ ⇒ 0 to 1 or -1 to 1

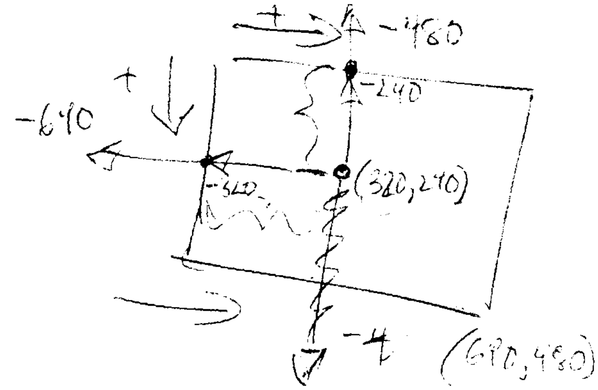
? $rand_L()$ ⇒ 0 to 1 or -1 to 1

? $rand_G()$ ⇒ 0 to 1 or -1 to 1

past best — Cognition ⇒ maximum visibility — Socialism ⇒ maximum $(\omega(t) - \gamma(t))_{ij}$

$$v_{ij}(M_{ij}(t+1)) = \varphi_R\, rand_R()\, M_{ij}(t) + \varphi_L\, rand_L()(M_{ij}(t) - x_{ij}(t)) + \varphi_G\, rand_G()((\omega_{ij}(t) - \gamma(t)) - x_{ij}(t))$$

new best



private void psoBehavior()
{
if (! stopped)
{
selectColor();

r = Math.random(); // $-1 \le r \le 1 \Rightarrow (0...1)*2+(-1)$
setVx(φ_k * r * getVx());
setVy(φ_k * r * getVy());

setV(φ_k * r * getV()); // $v_{t+1} = \varphi_k * r * v_t$

$\boxed{V_{iy}(t+1)}$
$\| $
$\varphi_k \, rand_k \, V_{iy}(t)$

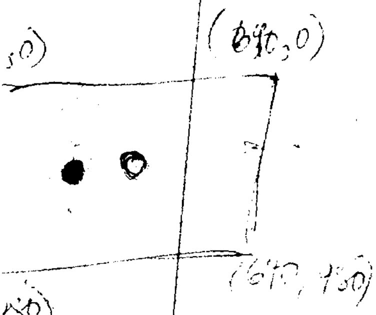
for (Pheromone p : pheromoneList)
{
setVx(getVx() + p.getTarget().getPosition().getX().distanceTo(this.getPosition().getX()));
setVy(getVy() + p.getTarget().getPosition().getY().distanceTo(this.getPosition().getY()));

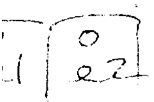
setV(getV() + p.getTarget().getPosition().distanceTo(this.getPosition()));

$\sum \varphi_i \, rand_i \, V_i(t)$

if (getVx > envDims.getWidth()) (640)
setVx(envDims.getWidth() - this.getPosition().getX());
if (getVx < 0)
setVx(-(this.getPosition().getX()));
if (getVy > envDims.getHeight()) (480)
setVy(envDims.getHeight() - this.getPosition().getX());
if (getVy < 0)
setVy(-(this.getPosition().getY()));

OR





Green margins included 09/10/06! NOTES: DON'T USE SIMPLE LANGUAGE

First Draft: Hybrid ACO/PSO Algorithm for Application to Autonomous Robot Swarm Architectures

Olorundamilola Kazeem 08/12/06

Olorundamilola Kazeem, Yan Wang, & Juan C. Müller

Abstract—**Presented in this paper is a new Hybrid ACO/PSO Architecture, a synergy of classic Ant Colony Optimization (ACO) and classic Particle Swarm Optimization (PSO), for robust solutions to complex, emergent problems in direct development with and application to an autonomous robot swarm. The intended autonomous robot swarm is composed of relatively simple, expendable autonomous robots of highly decentralized, self-organizing behaviors; which achieve global optimization through a posteriori, local behaviors on a priori tasks, i.e. cleaning hazardous chemical spills, gathering energy resources, attacking intruders, searching for and categorizing financial data, et cetera. This new Hybrid ACO/PSO Architecture adopts the feedback mechanism of ACO and the velocity/space correcting character of PSO to create a dynamic system to be implemented by a real autonomous robot swarm. A major thrust in the research was to design a simulation of the hybrid that conformed with the real world constraints. The accompanying simulation and results concretely supports the feasible application of the hybrid in a real world autonomous robot swarm system.**

Index Terms— **FOR THE 2006-2007 FORMAT INDEX TERMS ARE NOT REQUIRED.**

I. Introduction

Robust, dynamic, and adaptive computation methods are finding greater importance and attention, especially given the ever growing intricacies of design that are mostly and simply impossible to define on a basis of traditional computational methods. Traditional methods demand systems that are practically static over time, highly predictable, well-defined, and serialized. These hallmarks of traditional design become constraints and limitations on current and future engineering design thrusts, where the systems are dynamic, polymorphic, unpredictable, and non-serialized.

To serve these hard computational systems, the optimization methods of insect swarms, bird flights, and animal herds are adopted. The decentralized, emergent, self-organizing, collective attributes of these biological systems are adapted. The first of these ACO was developed in 1991 [1,2] and was applied to the computationally heavy Traveling Salesman Problem (TSP). Through a mechanism of feedback of simulated pheromones, and pheromone intensity processing, a semi-archive, the involved agents are steered toward local and global optimization. Another optimization technique PSO created in 1995 [5], has its foundation on the synchronization of bird flights. It works as a decision annealing device that converge to optimization in localities of particles and overall on the global front. Particles take consideration of past best position, neighbor's best position, and the global best; and determine the next optimal position, while avoiding collision and matching flight's speed and direction.

The application of these metaheuristics to autonomous robot swarms has significant benefits, however individually these methods ~~are lent~~ to problems and/or design challenges. Why? Well, direct translations of nature's designs are ~~either~~ impossible, infeasible, impractical, or ~~extremely~~ expensive. ~~Then~~ if nature's designs are going to be applied in an effective way, we must be able to take the ~~liberty~~ to ~~find the~~ appropriate analogies for our purposes, and even be so bold as to redesign minor to major aspects of the natural design. For example, a simple ACO application would reveal a bias towards local optima, because pheromone, the autocatalytic mechanism, by design is locally biased. A strict PSO adoption to an autonomous robot swarm would be difficult, since swarm entities are significantly blind over in reference to globalizing concerns. However, PSO is a sturdy decision processor for annealing premature convergence of particles in swarm situations. Thus, a new optimization technique, specifically tailored to the application of an autonomous robot swarm ~~had~~ to be designed. This new metaheuristic draws on the strengths of both systems: ~~ACO's~~ autocatalytic mechanism and ~~PSO's~~ decision or cognitive capabilities. One of the major challenges encountered was in designing the appropriate auto catalyst representation for the proposed autonomous robot swarm, a virtual entity-to-entity and entity-to-environment interaction mechanism. This mechanism is based on the pheromone drip trail of biological ants and other insects; However, an exact transpose of the biological mechanism would be vulnerable and expensive as a real implementation. The solution is introduced in Section II.

Section II introduces the ~~direct application of the~~ Hybrid ACO/PSO Architecture to a autonomous robot swarm, the design constraints and considerations, and the overall design redesigns and innovations. Section III explains the noise reduction protocols. Section IV details the framework and structure of the hybrid algorithm. Section V presents the simulation, and explains the virtual environment and real world constraints on the simulation. Section VI contains the results, the simulation data, and the hybrid simulation

serve dynamic, polymorphic, unpredictable, non-serialized systems

Manuscript received September 15, 2006. (Write the date on which you submitted your paper for review.) This work was supported by the National Science Foundation under Grant BS123456 (sponsor and financial support acknowledgment goes here).

F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (corresponding author to provide phone: 303-555-5555; fax: 303-555-5555; e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar. colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

Second Draft: Hybrid ACO/PSO Algorithm for Autonomous Robot Swarm Architectures

Ọlọrundamilọla Kazeem, Yan Meng, and Juan C. Müller 11 September 2006

Abstract—**Presented in this paper is a new Hybrid ACO/PSO Architecture, a synergy of classic Ant Colony Optimization (ACO) and classic Particle Swarm Optimization (PSO), for robust solutions to complex, emergent problems in direct development with and application to a autonomous robot swarm. The intended autonomous robot swarm is composed of relatively simple, expendable autonomous robots of highly decentralized, self-organizing behaviors; which achieve global optimization through a posteriori. local behaviors on a priori tasks, i.e. cleaning hazardous chemical spills, gathering energy resources, attacking intruders, searching for and categorizing financial data, et cetera. This new Hybrid ACO/PSO Architecture adopts the feedback mechanism of ACO and the velocity/space correcting character of PSO to create a dynamic system to be implemented by a real autonomous robot swarm. A major thrust in the research was to design a simulation of the hybrid that conformed with the real world constraints. The accompanying simulation and results concretely supports the feasible application of the hybrid in a real world autonomous robot swarm system.**

Index Terms— **FOR THE 2006-2007 FORMAT INDEX TERMS ARE NOT REQUIRED**

I. Introduction

Robust, dynamic, and adaptive computation methods are finding greater importance and attention, especially given the ever growing intricacies of design that are mostly and simply impossible to define on a basis of traditional computational methods. Traditional methods demand systems that are practically static over time, highly predictable, well-defined, and serialized. These hallmarks of traditional design become constraints and limitations on current and future engineering design thrusts, where the systems are dynamic, polymorphic, unpredictable, and non-serialized.

To serve these hard computational systems, the optimization methods of insect swarms, bird flights, and animal herds are adopted. The decentralized, emergent, self-organizing, collective attributes of these biological systems are adapted. The first of these ACO was developed in 1991 [1,2] and was applied to the computationally heavy Traveling Salesman Problem (TSP). Through a mechanism of feedback of simulated pheromones, and pheromone intensity processing, a semi-archive, the involved agents are steered toward local and global optimization. Another optimization technique PSO created in 1995 [5], has its foundation on the synchronization of bird flights. It works as a decision annealing device that converge to optimization in localities of particles and overall on the global front. Particles take consideration of past best position, neighbor's best position, and the global best; and determine the next optimal position, while avoiding collision and matching flight's speed and direction.

The application of these metaheuristics to autonomous robot swarms has significant benefits, however individually these methods are lent to problems and/or design challenges. Why? Well, direct translations of nature's designs are either impossible, infeasible, impractical, or extremely expensive. Then if nature's designs are going to be applied in an effective way, we must be able to take the liberty to find the appropriate analogies for our purposes, and even be so bold as to redesign minor to major aspects of the natural design. For example, a simple ACO application would reveal a bias towards local optima, because pheromone, the autocatalytic mechanism, by design is locally biased. A strict PSO adoption to an autonomous robot swarm would be difficult, since swarm entities are significantly blind over in reference to globalizing concerns. However, PSO is a sturdy decision processor for annealing premature convergence of particles in swarm situations. Thus, a new optimization technique specifically tailored to the application of an autonomous robot swarm had to be designed. This new metaheuristic draws on the strengths of both systems: ACO's autocatalytic mechanism and PSO's decision or cognitive capabilities. One of the major challenges encountered was in designing the appropriate auto catalyst representation for the proposed autonomous robot swarm, a virtual entity-to-entity and entity-to-environment interaction mechanism. This mechanism is based on the pheromone drip trail of biological ants and other insects; However, an exact transpose of the biological mechanism would be vulnerable and expensive as a real implementation. The solution is introduced in Section II.

Section II introduces the direct application of the Hybrid ACO/PSO Architecture to a autonomous robot swarm, the design constraints and considerations, and the overall design redesigns and innovations. Section III explains the noise reduction protocols. Section IV details the framework and

Manuscript received September 15, 2006. (Write the date on which you submitted your paper for review.) This work was supported by the National Science Foundation under Grant BS123456 (sponsor and financial support acknowledgment goes here).

O. Kazeem is with the New York University, New York, NY 80305 USA (corresponding author to provide phone: 303-555-5555; fax: 303-555-5555; e-mail: ok288@nyu.edu).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar. colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).