Additional Files: Appendix of ALVEC related codes

Appendix A. Stability condition for Lotka-Volterra Model:

$$\frac{dP}{dt} = \alpha P - \beta PQ \tag{A.1}$$

$$\frac{dQ}{dt} = \delta PQ - \gamma Q \tag{A.2}$$

The stability of this model is attained, when there is no growth rate for both P, Q.

$$\begin{aligned} \alpha P - \beta PQ &= 0 \\ => P\left(\alpha - \beta Q\right) &= 0 \end{aligned}$$

Either P=0 or $\alpha - \beta Q$=0. P=0 is trivial. Hence

$$\begin{aligned} \alpha - \beta Q &= 0 \\ => \alpha = \beta Q; \quad \frac{\alpha}{\beta} &= Q \end{aligned}$$

if $\beta = 1$, than $\alpha = Q$
Similarly,

$$\begin{aligned} \delta PQ - \gamma Q &= 0 \\ => Q\left(\delta P - \gamma\right) &= 0 \\ => \gamma = \delta P; \quad \frac{\gamma}{\delta} &= P \end{aligned}$$

if $\delta = 1$, $\gamma = P$. Therefore, stability of the proposed model occurs when $\alpha = Q$ and $\gamma = P$. The stationery point is evaluated as $(\frac{\alpha}{\beta}, \frac{\gamma}{\delta})$, mentioned in Fig.2 (Main File).

Appendix B. Matlab Code for Fig. 4,5 and 6 (Main File)

$span = 0 : 0.1 : 100;$

$[t, y] = ode45(@Lotka, span, [60; 80]);$

$plot(t, y(:, 1),' -o', t, y(:, 2),' -o')$

$function \quad dydt = Lotka(t, y)$

$dydt = [20 * y(1) - y(1) * y(2); -20 * y(2) + y(1) * y(2)];$

end

Appendix C. R Code for data fitting: Fig.29 (Main File)

The following R code is used to generate the plot:

```
library(plotrix)
library(ellipse)
data = read.csv("Merged.csv")
data1 = read.csv("sheet2.csv"); \\ data set from cloudsim results of LV runs;
predator increasing-prey decreasing \\
data2 = read.csv("sheet1.csv"); \\ data set from cloudsim results of LV runs;
predator decreasing-prey increasing \\
x=data$Cloudlet.Number
y=data$VM.Number
x1= data1$Cloudlet
y1=data1$VM
x2=data2$Cloudlet
y2=data2$VM
#ellipse fitting with 95\% confidence level
eli1 = ellipse(cor(x1,y1),scale=c(sd(x1),sd(y1)),
centre=c(mean(x1), mean(y1)), level = 0.95, npoints = 250)
eli2 = ellipse(cor(x2,y2),scale=c(sd(x2),sd(y2)),
centre=c(mean(x2), mean(y2)), level = 0.95, npoints = 250)
#Calculate the center of ellipse
        [c1x,c2y] = c(mean(eli1[,1]), mean(eli1[,2]))
        [c2x,c2y] = c(mean(eli2[,1]), mean(eli2[,2]))
    sink()
```

```r
jpeg("plotfit.jpeg")
plot(x,y)
draw.ellipse(118.21488,31.59861 , a = 90, b = 25)
draw.ellipse(29.98445,55.116337, a = 20, b = 35)
dev.off()
```