Modelling source and sink dynamics

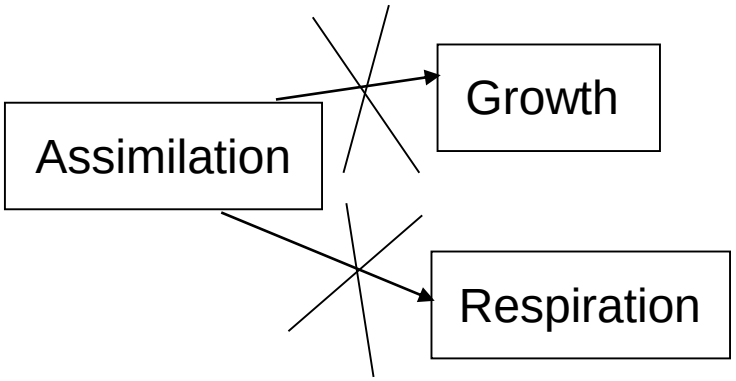
Loss due to maintenance respiration

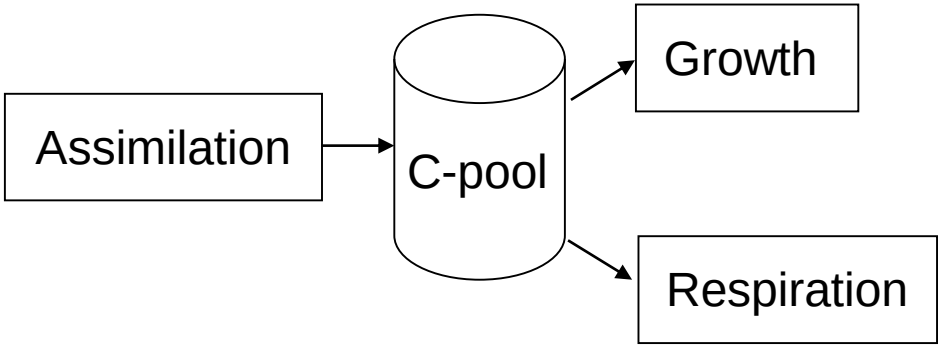
- Respiratory losses implied in the calculated growth rate
- Separation of gross growth rate and respiration losses possible.
- Simplest form: respiratory loss ~ accumulated biomass
- Value of 'specific' (i.e. mass-related) maintenance coefficient (*m*) between 0.01 and 0.03 g g^{-1} d^{-1}
- r_m (0.1 – 0.2 g g^{-1} d^{-1}) ≈ 10 * m

for a constant $c_{m,g}$, growth can no longer be linear, because with increasing biomass, a growing portion of the absolute growth rate is used for maintenance purposes.

Assimilate flow and respiration

- Loss of carbohydrates by some form of respiration
- No perfect match in timing of acquisition of carbohydrates and of their utilisation for growth and respiration → temporary storage necessary

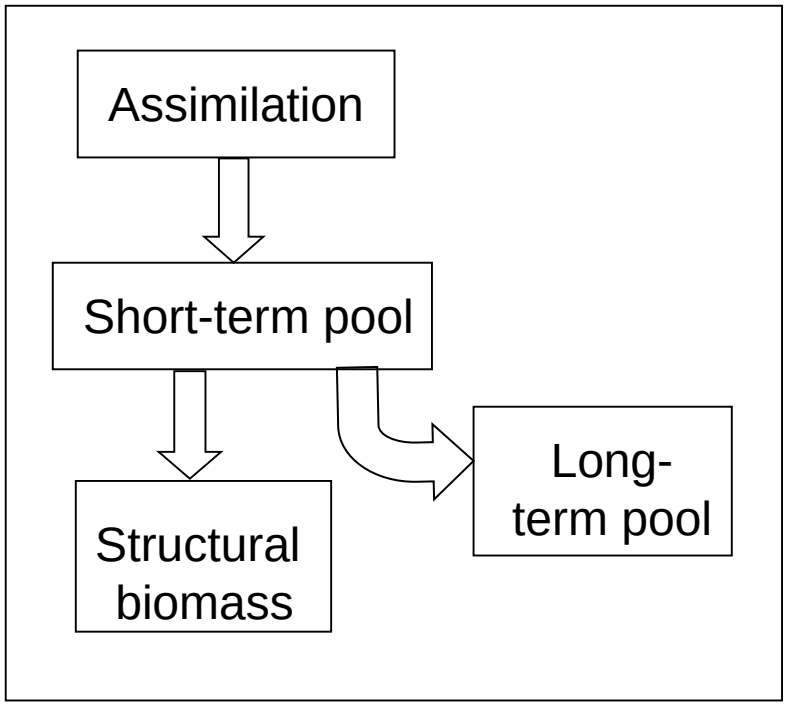




- Follow-up of diurnal course of (C-) flows: distinction within the CH_2O-pools between:
  - short-term reservoir (e.g., starch granules)
  - structural biomass (cellulose, lignin, ...)
- or lump it together to total plant biomass.

Assimilate flow and respiration

- reserve pools
- short term: diurnal course of photosynthesis
- long-term: reserve pools (sugar in the stalks of grasses and cereal crops, starch in potato tubers), used to store material during a longer period, used later for, e.g., grain filling or formation of new tillers.
- In crop model, daily reserve pool placed between source (assimilation) and sinks (maintenance and growth)

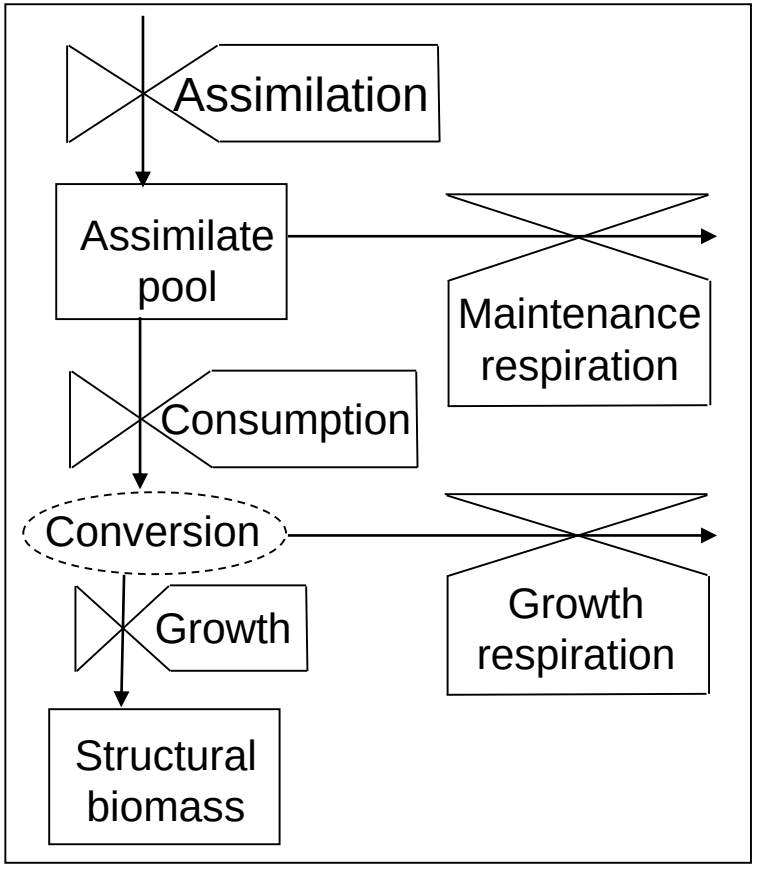


Assimilate flow and respiration

- Coefficients are related by conservation of carbon:

$$(12/30) \cdot G = C + (12/44) \cdot \Phi \qquad \left[gC \cdot g^{-1}DM\right]$$

- Inverse of *G* (ca. 0.7) often referred to as the conversion factor.
- Generally, value of these coefficients depends on the type of end product.



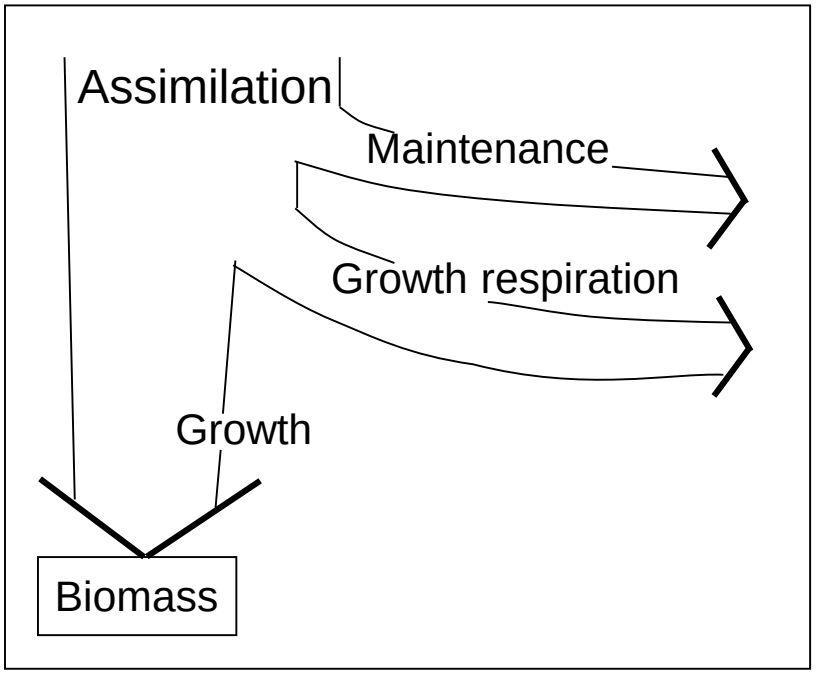
C: Average carbon content

G: Glucose requirement for growth

Φ: CO_2 production during growth

Assimilate flow and respiration

- In many models diurnal course of the assimilate buffer is disregarded → Time increment for integration = 1 day, i.e. assimilation immediately transferred to maintenance and consumption



- priority of maintenance over growth.
- priority emerges as a result of the *absence* of feedback of assimilate content on maintenance, in contrast to the *presence* of feed-back on growth.

•: Growth and maintenance draw simultaneously on assimilate pool, no priority. If assimilate suppy not sufficient gradual reduction of assimilate level and growth rate.

- maintenance process continues whatever assimilate level.

- Appropriate long-term description: growth will gradually adjust to flow of assimilates remaining after deduction of maintenance requirement.

Outline

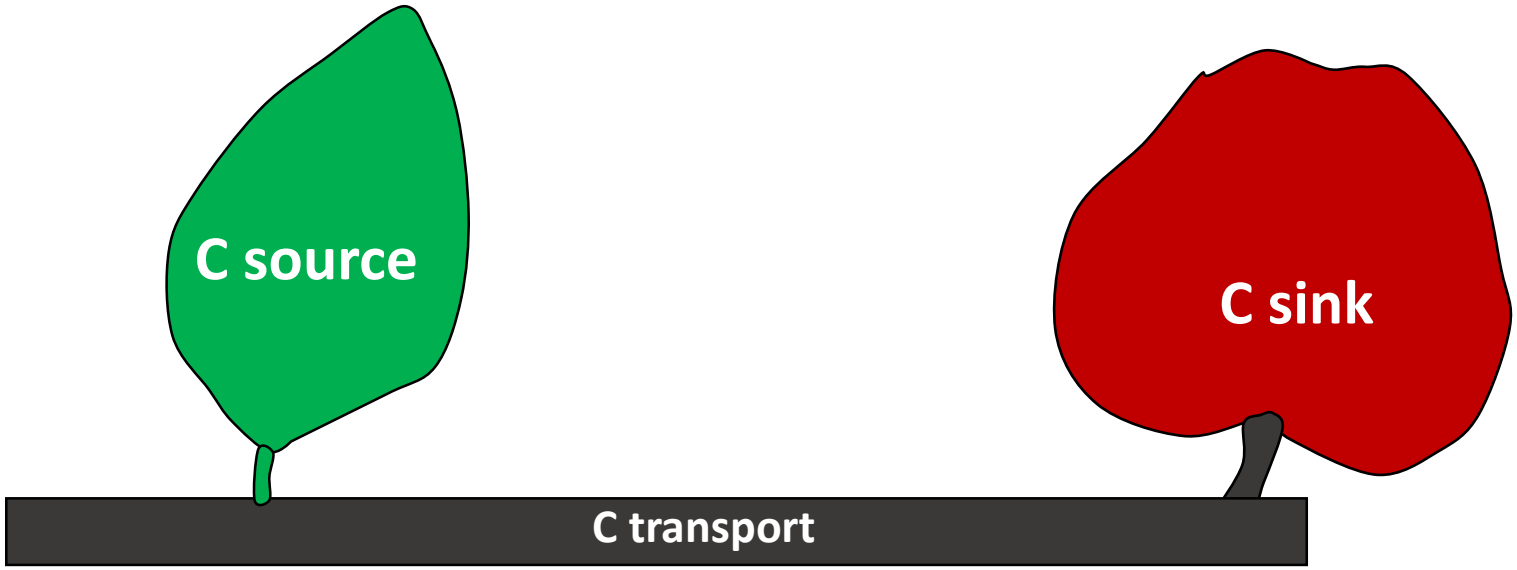
- Introduction
- Branch architecture and C production
- Branch architecture and C partitioning

Introduction

- Factors affecting fruit growth

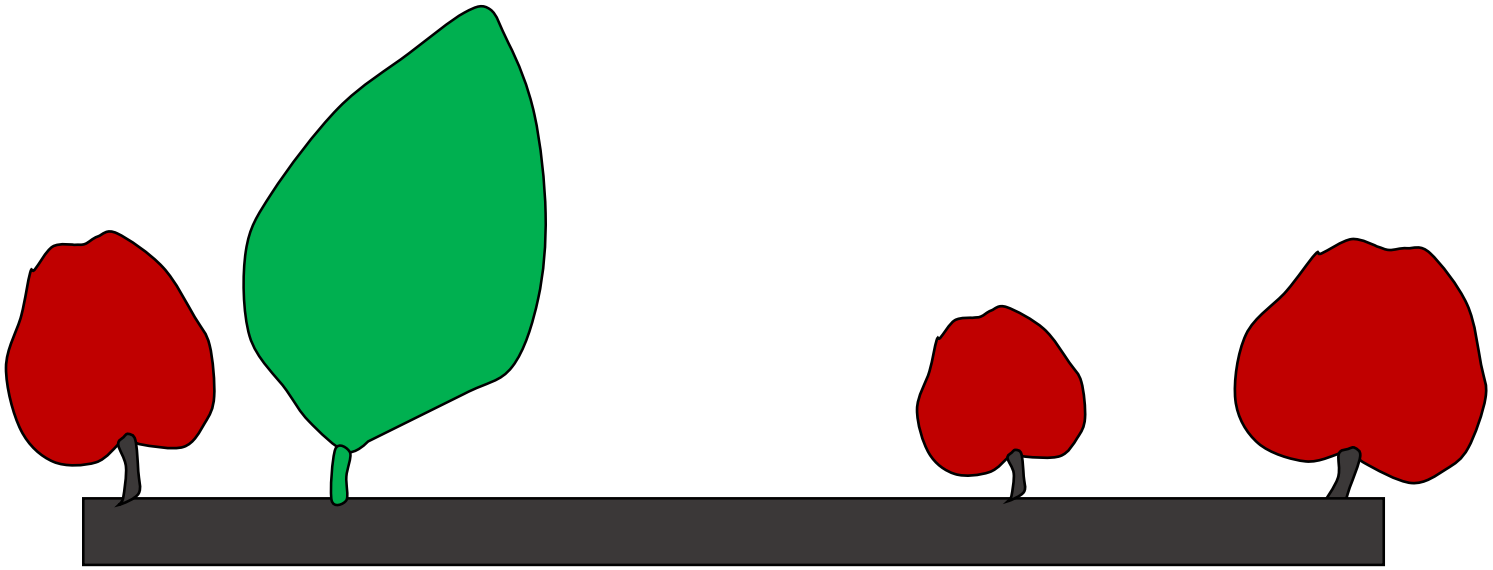
Introduction

- Factors affecting fruit growth



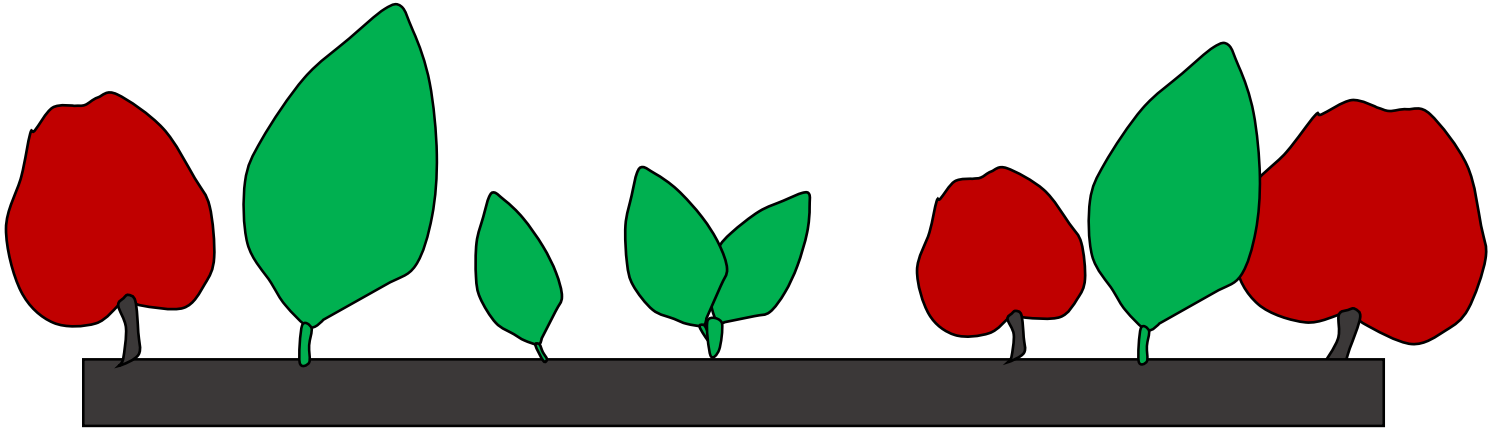
Introduction

- Factors affecting fruit growth



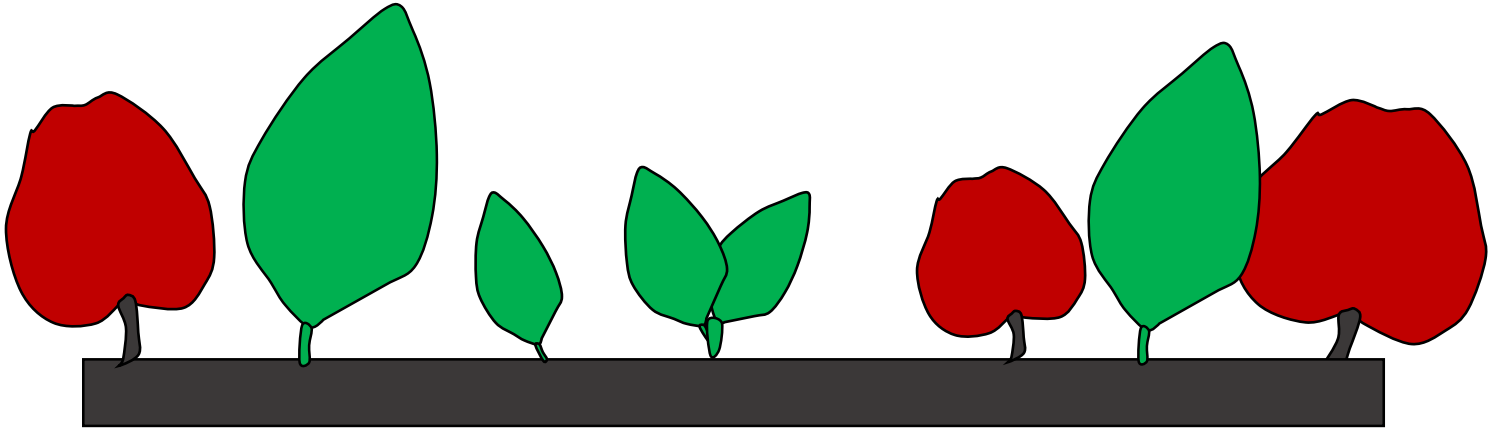
Introduction

- Factors affecting fruit growth



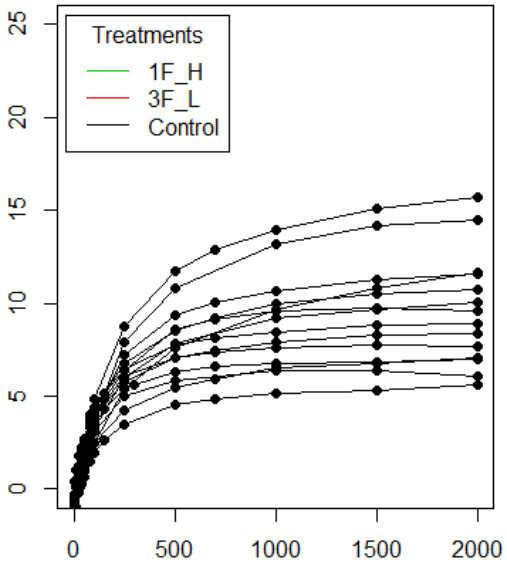
Introduction

- Factors affecting fruit growth

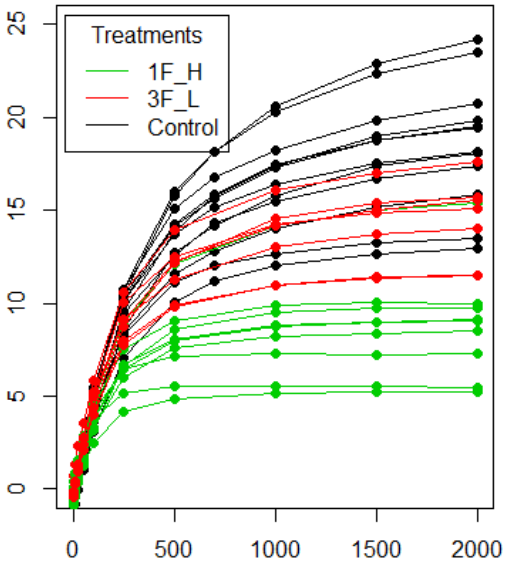


- Architecture → distribution of sources and sinks in space & time

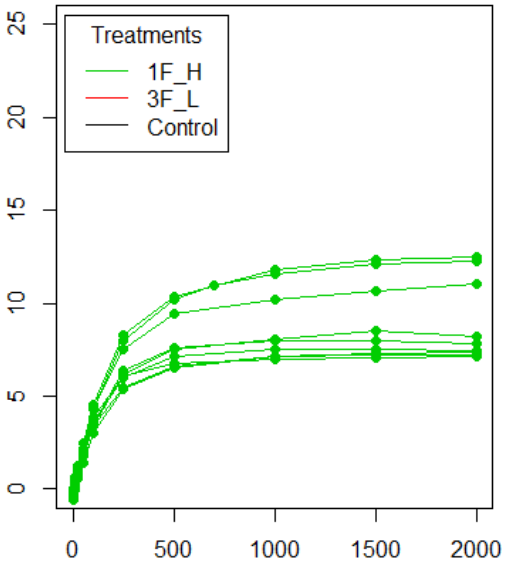
Light response curves



Rosette



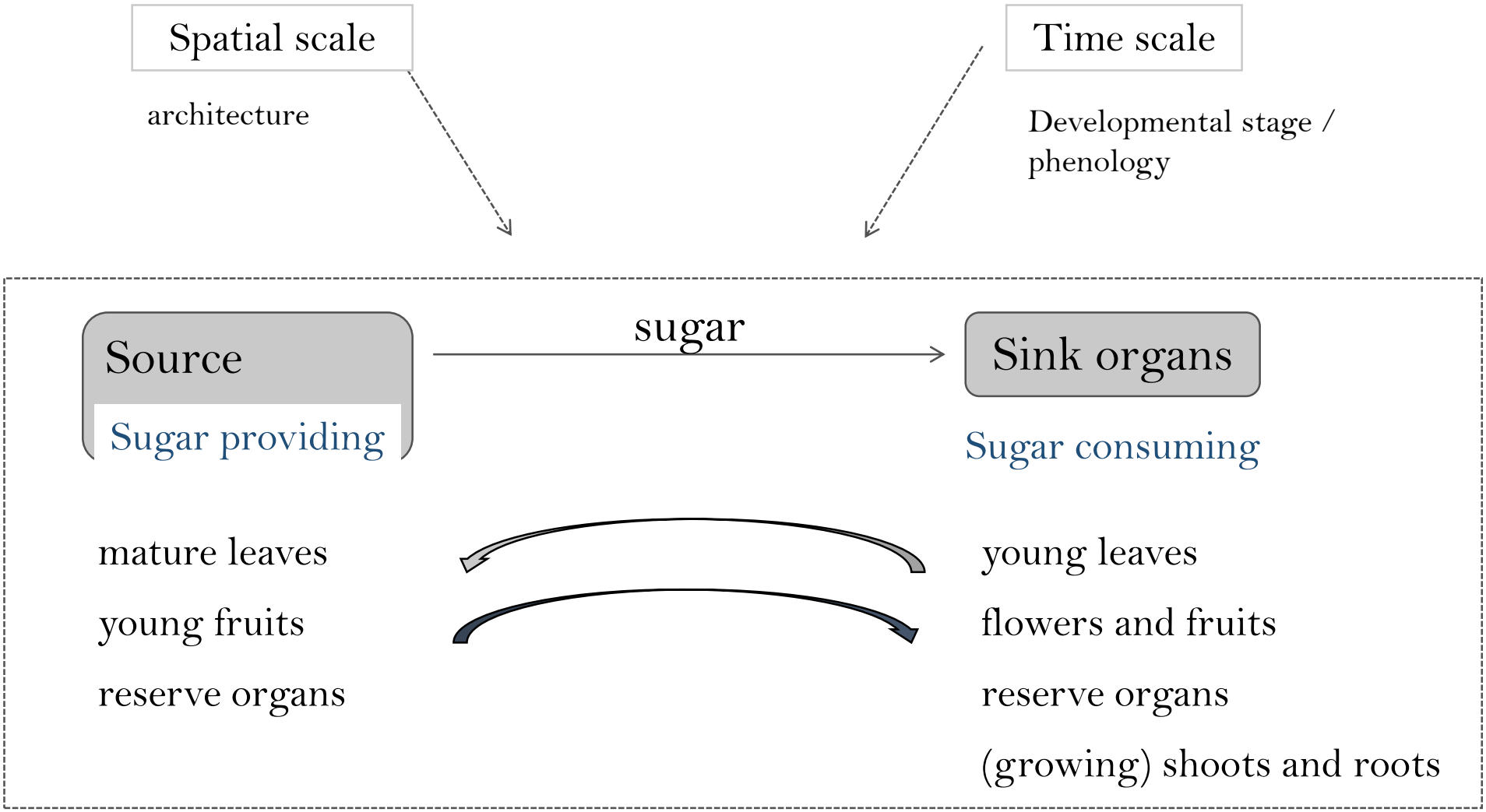
Bourse shoot



Vegetative Shoot

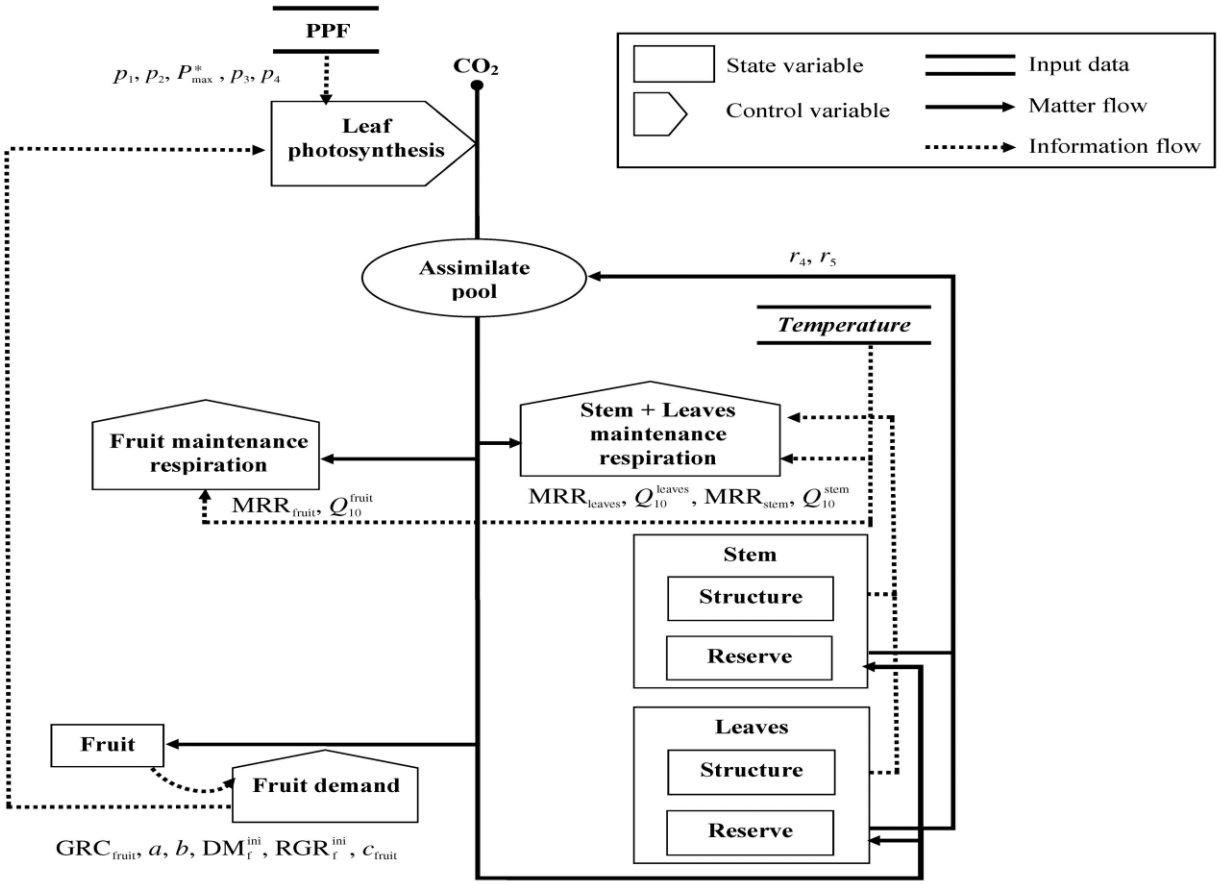
Bairam, 2017

Source and Sink Organs



Carbohydrate partitioning depends on organ demand and priority rules. The priority rules used for mango were:

1. maintenance of the system
2. reproductive growth
3. accumulation of reserves in the leaves and then in the stem. When the potential fruit demand is higher than carbohydrates supplied by photosynthesis, the fruit may obtain assimilates from leaf and stem reserves.



Lechaudel et al. 2005

- Plant **central pool** ← refilled at each time step by evaluating hourly assimilate stock of each leaf
- Each organ carries out maintenance and growth respiration.
- Growth modelled using the derivative of a logistic function, according to the requirements of each organ type.
  - Organ assimilate requirement (for growth and maintenance) modelled according to **relative sink strength (RSS)** concept (Marcelis et al., 1998):
    - 1) potential growth rate (PGR) determined using derivative of sigmoid growth function (e.g. logistic).
    - 2) For a given organ type (fruit, internode, root, leaf): RSS = individual $PGR_{organ}/\Sigma PGR_{all\ organs,\ f|i|r|l}$ of the same type.
    - Product of RSS and a percentage of the central pool (75 percent of the central pool allocated to fruit, Marcelis 1994) = amount of assimilates that can be allocated to a given organ at a given time.

Carbon management at plant individual level:

```
module Individual (int indiID, int age, int treatment) {
    float carbonPool = 0.25;
    float carbonPoolRoot = 0;
    float carbonPoolLeaf = 0;
    float carbonPoolInt = 0;
    float MRDemandRoot = 0;
    float MRDemandInt = 0;
    float MRDemandLeaf = 0;
    float GRDemandRoot = 0;
    float GRDemandInt = 0;
    float GRDemandLeaf = 0;
```

```
// Gestion des ressources carbon,es
void dimCarbonPool(float amount) {
        if(this.carbonPool> 1/(1-RESERVE_CCP)*amount) {
        this.carbonPool -= amount;}
        else {
        amount=(1-RESERVE_CCP)*this.carbonPool; // RESERVE_CCP défini
        this.carbonPool -= amount;}
```

```
  float getCarbonPool() {
      carbonPool += sum((* lf:Leaf, (lf[pid]==this[indiID]) *)[as])*44.01e-6
      // modifié 7/5/24: distinction entre les deux traitements
      *3600
  * (180.162/264.06)
  / 1000.0;
  ccp[this[indiID]-1] = carbonPool;
      return carbonPool;
  }
```

Carbon management at plant individual level:

```
//carbon pool for leaves
void emptyCarbonPoolLeaf() {
    carbonPoolLeaf = 0;
}

float getCarbonPoolLeaf() {
    carbonPoolLeaf = carbonPool*PARTITION_LEAF;
    return carbonPoolLeaf;
}

 // maintenance respiration rate for leaves
 float getMRDemandLeaf() {
     MRDemandLeaf = sum((* Leaf *).getMR());
     return MRDemandLeaf;
 }

// growth respiration rate for leaf
float getGRDemandLeaf() {
    GRDemandLeaf = sum((* Leaf *).getGR());
    return GRDemandLeaf;
}
```

→ Analogously for roots and stems!

```
//potential growth rate (PGR) of internodes
float getPGRInt() {
    PGRInt = sum((* Internode *).getPGR());
    return PGRInt;
}
```

→ Analogously for roots!

```
//potential growth rate (PGR) of leaves
float getPGRLeaf(float nitrogen) {
    PGRLeaf = sum((* Leaf *).getPGRLeaf(this.nitrogen));
    return PGRLeaf;
}
```

```
//total respiration demand leaf
float getTotalRespDemandLeaf() {
    totalRespDemandLeaf = this.getGRDemandLeaf() + this.getMRDemandLeaf();
    return totalRespDemandLeaf;
}
```

→ Analogously for roots and stems!

```
//total sugar demand internode
float getTotalSugarDemandInt() {
    totalSugarDemandInt = this.getTotalRespDemandInt() + this.getPGRInt();
    return totalSugarDemandInt;
}
```

→ Analogously for roots and leaves (but also depending on N)!

```
float getDW()
{
    DW = 0.05*(sum((* itn:Internode, (itn[pid]==this.indiID) *).getDW()) +
        sum((* rt:Root, (rt[pid]==this.indiID) *).getDW()) +
        sum((* lf:Leaf, (lf[pid]==this.indiID) *).getDW()));
    return DW;
}
```

Definition of update carbon pool of individual plant…

```
public void updateCP() {
    getNitrogenReserve(); //update nitrogen uptake
    getSulphateReserve();  //update sulphate uptake
    getPhosphateReserve();  //update phosphorus uptake
    getMagnesiumReserve();  //update magnesium uptake
    getCalciumReserve();  //update calcium uptake
    getPotassiumReserve();  //update potassium uptake
    getCarbonPool();  //update source
    getCarbonPoolRoot();//update source
    getMRDemandRoot(); getGRDemandRoot(); getPGRRoot();
    getTotalRespDemandRoot(); getTotalSugarDemandRoot();
    getCarbonPoolLeaf();//update source
    getMRDemandLeaf(); getGRDemandLeaf(); getPGRLeaf(this.nitrogen);
    getTotalRespDemandLeaf(); getTotalSugarDemandLeaf();
    getCarbonPoolInt();//update source
    getGRDemandInt(); getPGRInt();
    dimCarbonPool(getTotalSugarDemandRoot() +
    getTotalSugarDemandLeaf() + getTotalSugarDemandInt());
}
```

… invoked in the main rgg (sunflower)

```
protected void CPool()
[
    //Dummy(t,r) ==> Dummy(t+1,r);
    ind:Individual ::>
        { ind[age]++; ind.updateCP();
```

Updating at organ level (example leaf, analogously in stems and roots!)

```
float getSurface()
{
    surface = length*width*LEAF_FF;
    return surface;
}

float getFW()
{
    FW = surface*90.279;
    return FW;
}

float getDW()
{
    DW = getFW()*DW_RATE_LEAF;
    return DW;
}

float getMR()
{
    MR = MR_COEFF_LEAF*getDW();
    return MR;
}
```

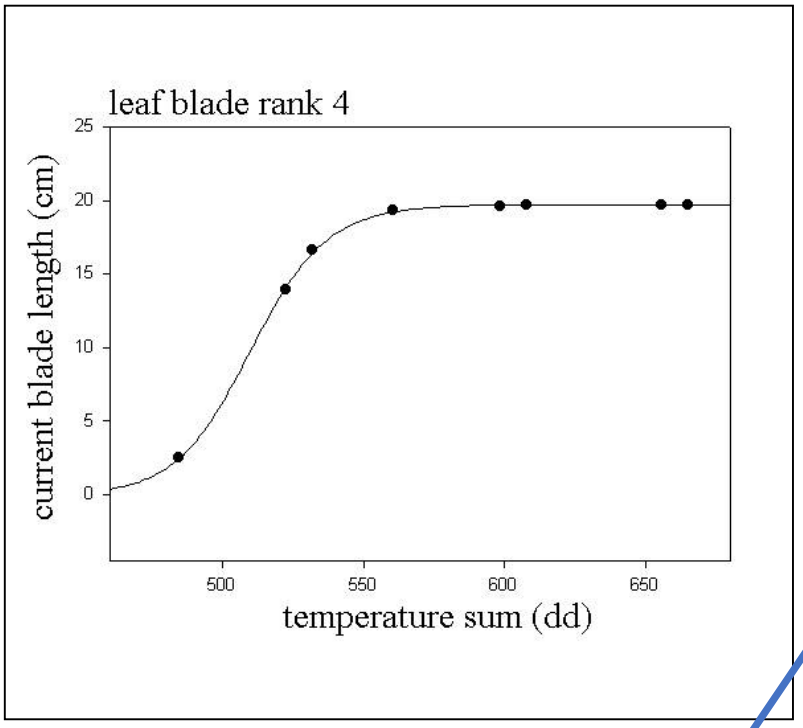
```
float getPGRLeaf(float nitrogen)
{
    PGRLeaf = logistic(NITROGEN[ACTIVETREATMENT[pid-1]]*0.12, age,
        150-NITROGEN[ACTIVETREATMENT[pid-1]]*5, NITROGEN[ACTIVETREATMENT[pid-1]]*0.01);
    return PGRLeaf;
}

float getAGRLeaf(float totalSugarDemand, float Nitrogen)
{
    AGRLeaf = this.getPGRLeaf(Nitrogen)/totalSugarDemand;
    return AGRLeaf;
}

float getGR()
{
    GR = GR_COEFF_LEAF*PGRLeaf;
    return GR;
}
```

These methods are called when the organ in question (e.g. Leaf) is updated (method absorbAndGrow()) in sunflower.rgg:

```
lf.getMR(); lf.getGR(); lf.getPGRLeaf(ind[nitrogen]);
lf.getFW(); lf.getDW(); lf.getAGRLeaf(ind.getTotalSugarDemandLeaf(),ind[nitrogen]);
```



Logistic function with 3 parameters:

C = maxdim

M = phylloM

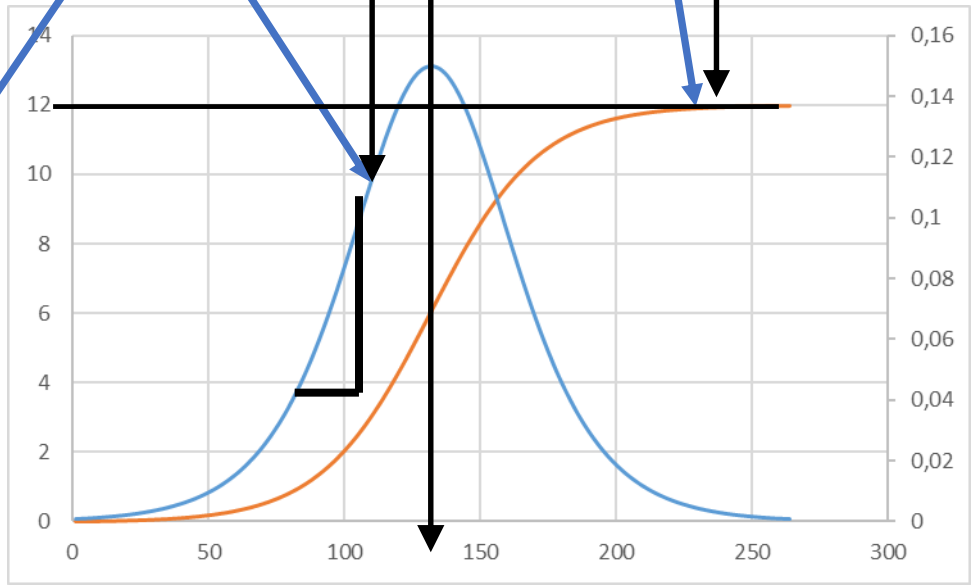
B = slope

and state variable

t = time (age of the growing organ)

Derivative (growth rate)

Integral (development of organ size/length…) in time



$$\frac{dy}{dt} = \frac{B \cdot C \cdot e^{-B(t-M)}}{(e^{-B(t-M)} + 1)^2}$$

```
public static float logistic (float maxdim, int time, float phylloM, float slope)
{
    return (slope*maxdim*Math.exp(-slope*(time-phylloM)))/
        ((Math.exp(-slope*(time-phylloM))+1)**2);
}
```