Contents lists available at ScienceDirect

Renewable and Sustainable Energy Reviews

journal homepage: http://www.elsevier.com/locate/rser

Designing a sustainable closed-loop supply chain network for walnut industry

Amirhossein Salehi-Amiri [a], Ali Zahedi [b], Navid Akbapour [b], Mostafa Hajiaghaei-Keshteli [b,*]

[a] Department of Systems Engineering, École de Technologie Supérieure (ÉTS), University of Quebec, Montreal, Canada
[b] Tecnologico de Monterrey, Escuela de Ingeniería y Ciencias, Puebla, Mexico

Sustainability

Economic

Environmental

Social

Closed-loop Supply chain

Sustainability

Customer Loyalty

Reduced Costs

Introduction

ASC

Nuts/Walnut

Usages & Benefits

Fig 1. Trend of publications in the field of agriculture supply chain: (a) number (b) publication category.

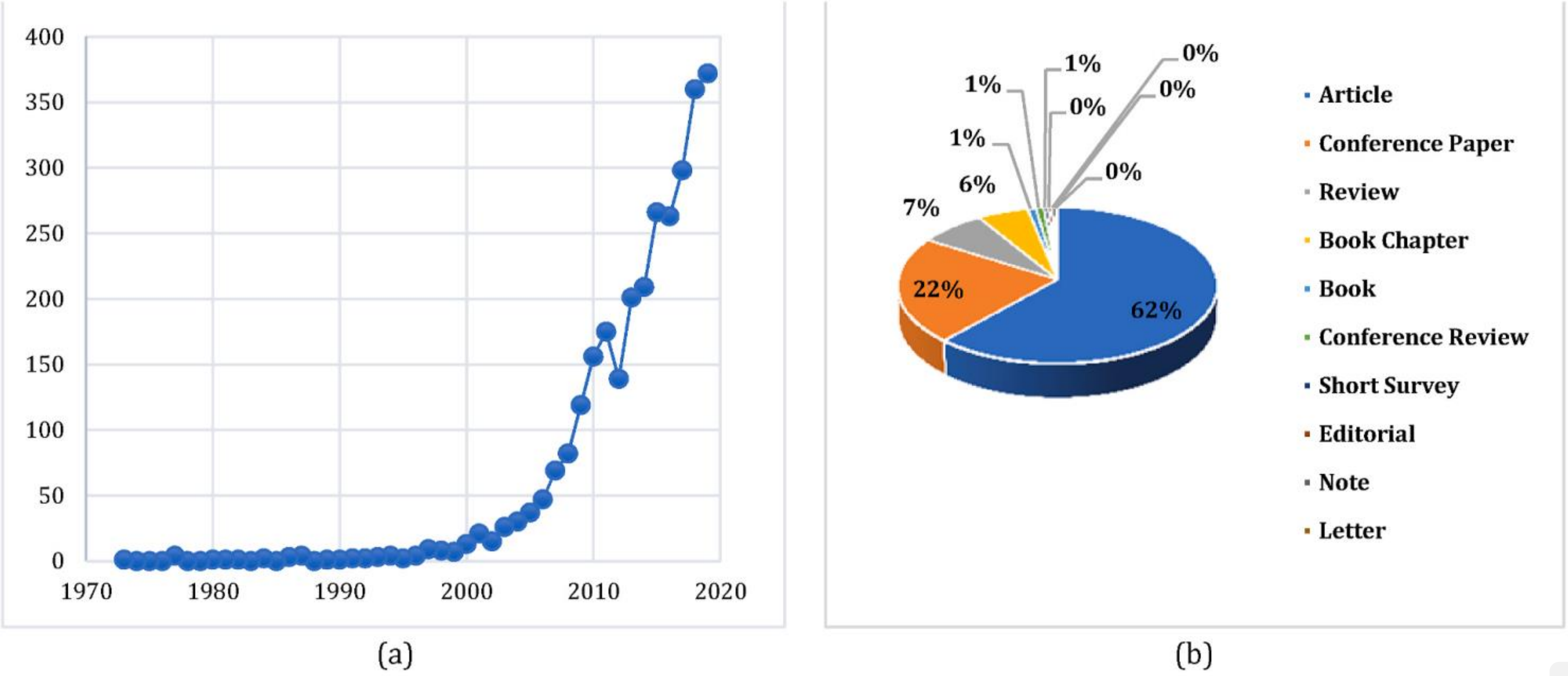
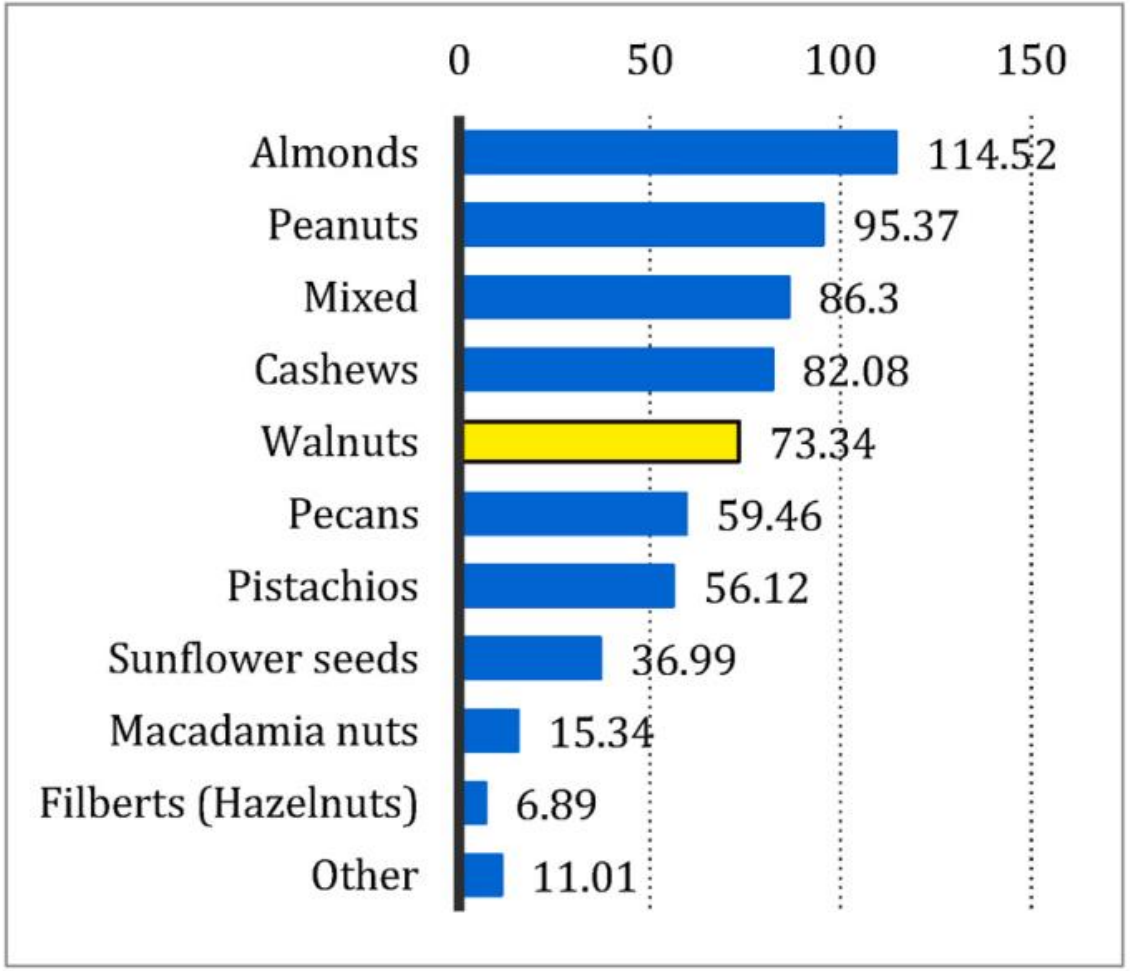
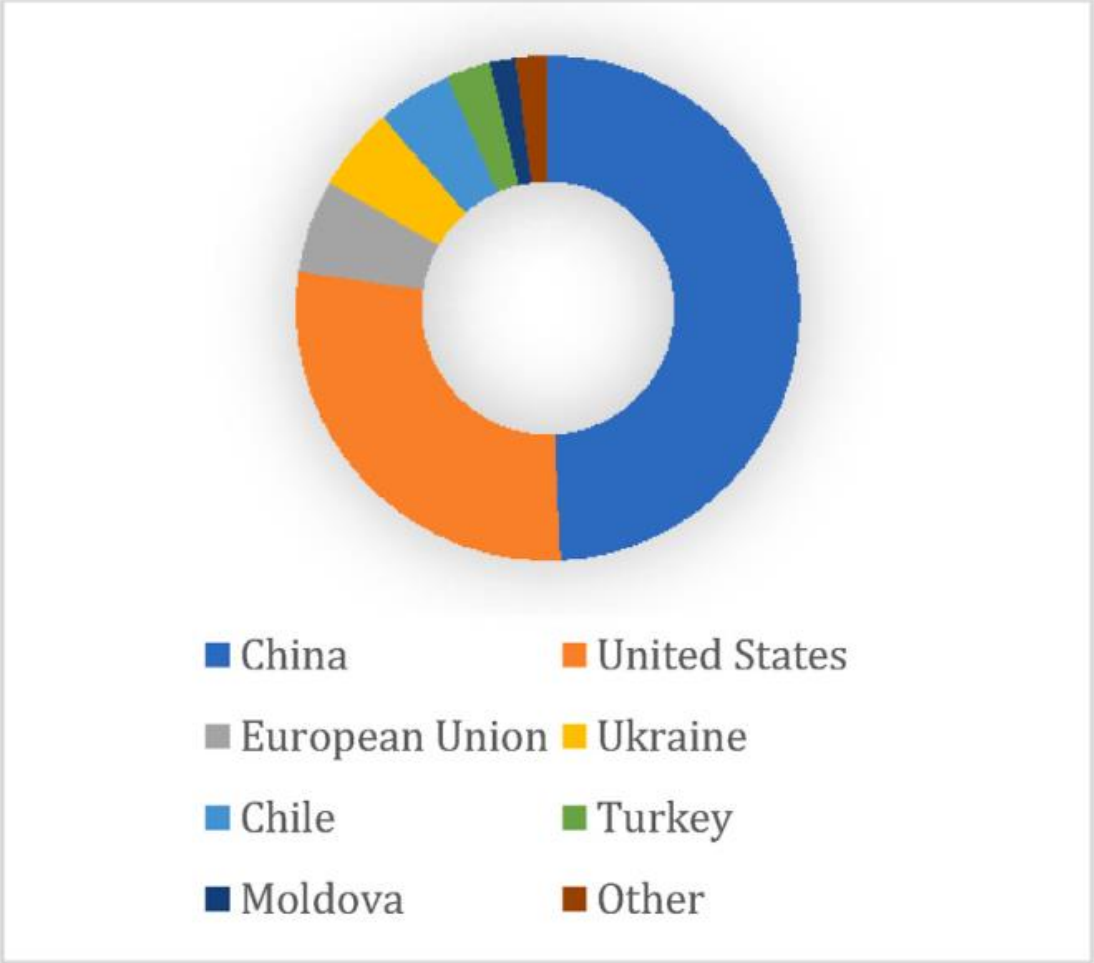


Fig 2. Walnut production share (a) US consumption (b) global production share.



(a)



(b)

Fig 5. Walnut constituents and their usage.

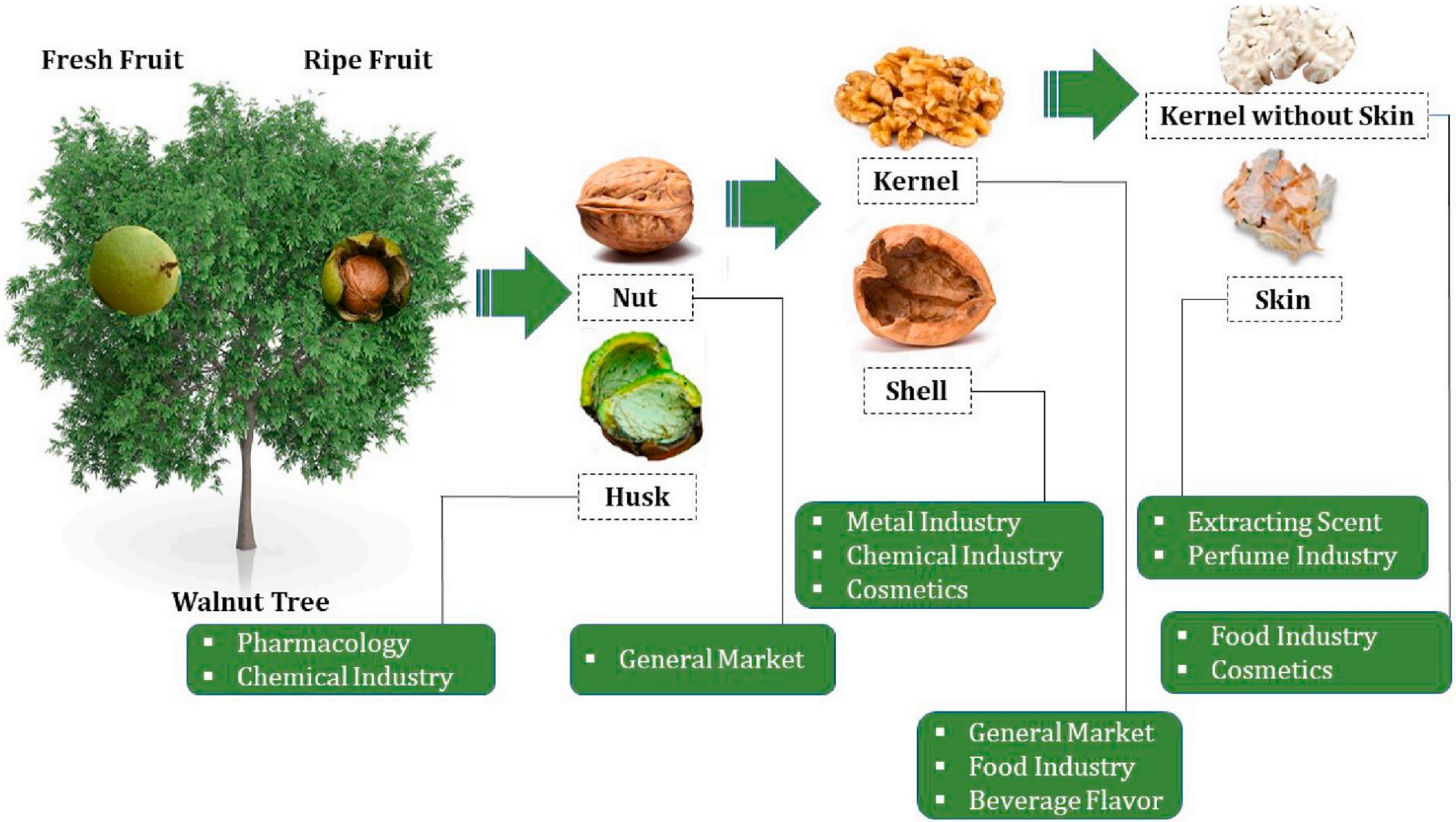


Fig 6. Walnut constituents and their usage.

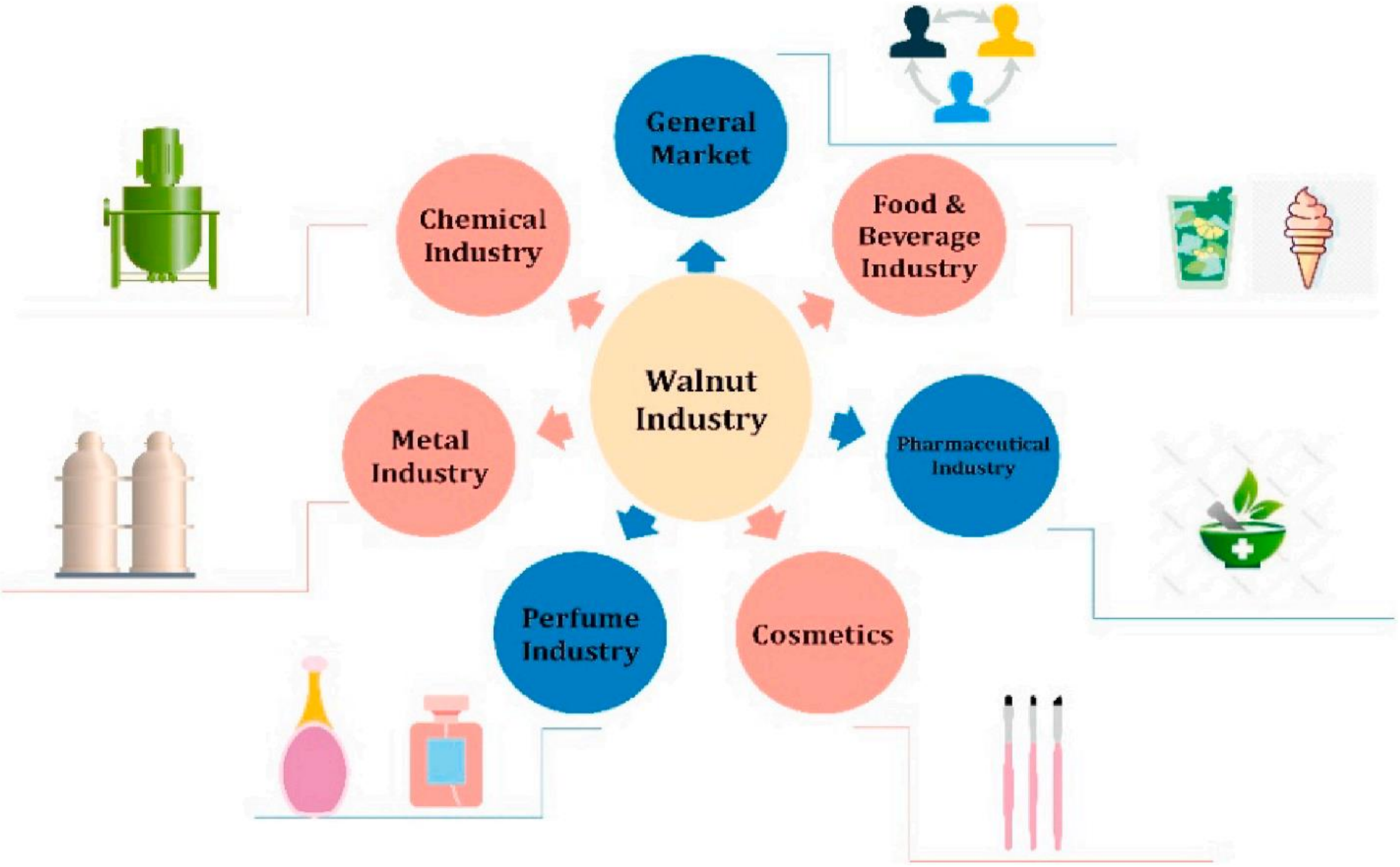
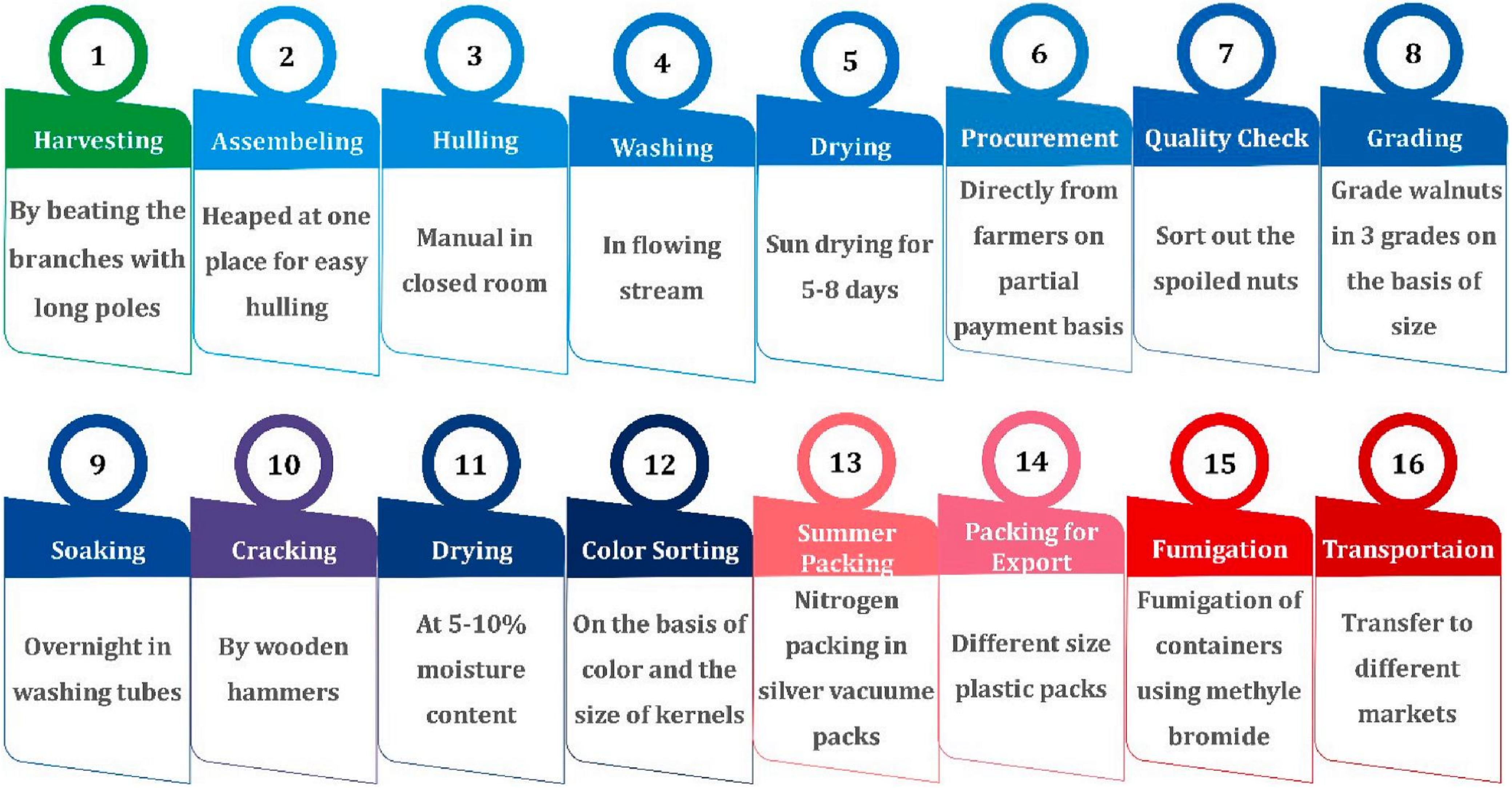


Fig 7. The main process for walnut value chain



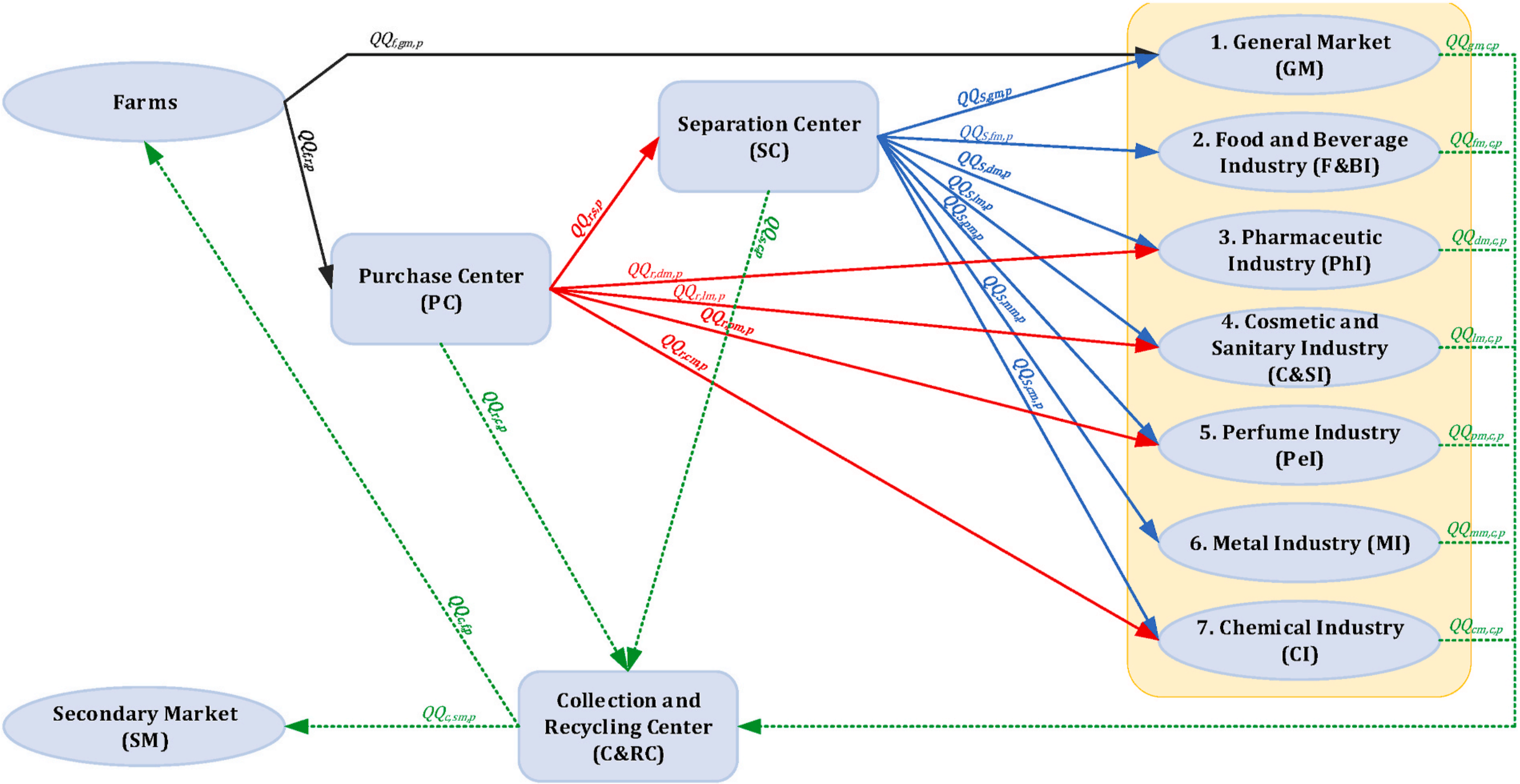
Mathematical Formulation

Network

Model

Parameters

Fig 9. Structure of the walnut supply chain network.



Solution

Approach

Meta-Heuristics

SA - SEO - KA

Hybrid Meta-Heuristics

HKSA - HKSEO

Encoding

Decoding

Section 5					
c		$sm+f$			
0.81	0.04	0.33	0.62	0.26	0.25
1	6	3	2	4	5

Fig 13. The pseudo-code of the SA algorithm.

```
Solution: initial random solution generation
T: initial temperature
Value: solution evaluation
While ceasing condition not confirmed Do
    for B= 1: Maximum sub iteration Do
        New solution: neighborhood solution generation
        New Value: New solution evaluation
            if new value < value
                solution=new solution
            value=new value
        else
            Δ= new value – value
            acceptation of new solution probability=e^(Δ/T)
        end if
    end for
reduce T
end while
```

Fig 13. The pseudo-code of the SEO algorithm.

```
Initialize an attacker and defender
It=0;
while It <Maxit
    Do training and retraining;
    Num_attack=0;
    while Num_attack <Max_attack
        Spot an attack;
        Check the boundary;
        Respond to attack,
        if the Objective Function (OF) of the defender
        is lower than the attacker
            Exchange the defender and attacker position;
        end if
        Num_attack=Num_attack+1;
    end while
    Create a new solution as a defender,
    It=It+1;
end while
Return the attacker
```

Fig 15. The pseudo-code of the KA algorithm.

Initialize feasible Keshtels population randomly
Calculate the fitness
Sort Keshtels in three types: N_1 as the best, N_2, and N_3 as the worst solutions
for i = 1 to MaxIt
for each N_1
Select a chromosome a_i from N_1 population
Find the nearest neighbor to an in the populations
Determine the new solution b_i by finding distance between this best Keshtel a_i and between neighbors.
Calculate the cost function of the new solution
If new solution a_i is better than the b_i, then $a_i = b_i$
else
Considering half the radius generates two solutions b_{i1} and b_{i2}
end if
If new solution b_{i1} or b_{i2} is better than a_i then replace it with a_i
else
use the final generated solution as a_i
end if
end for
for each N_2
selects two chromosomes randomly
move the chromosome of the population N_2 according to two selected chromosomes
end for
Generates N_3
chromosomes randomly
merge population N_1 , N_2 and N_3
sort the population
end for

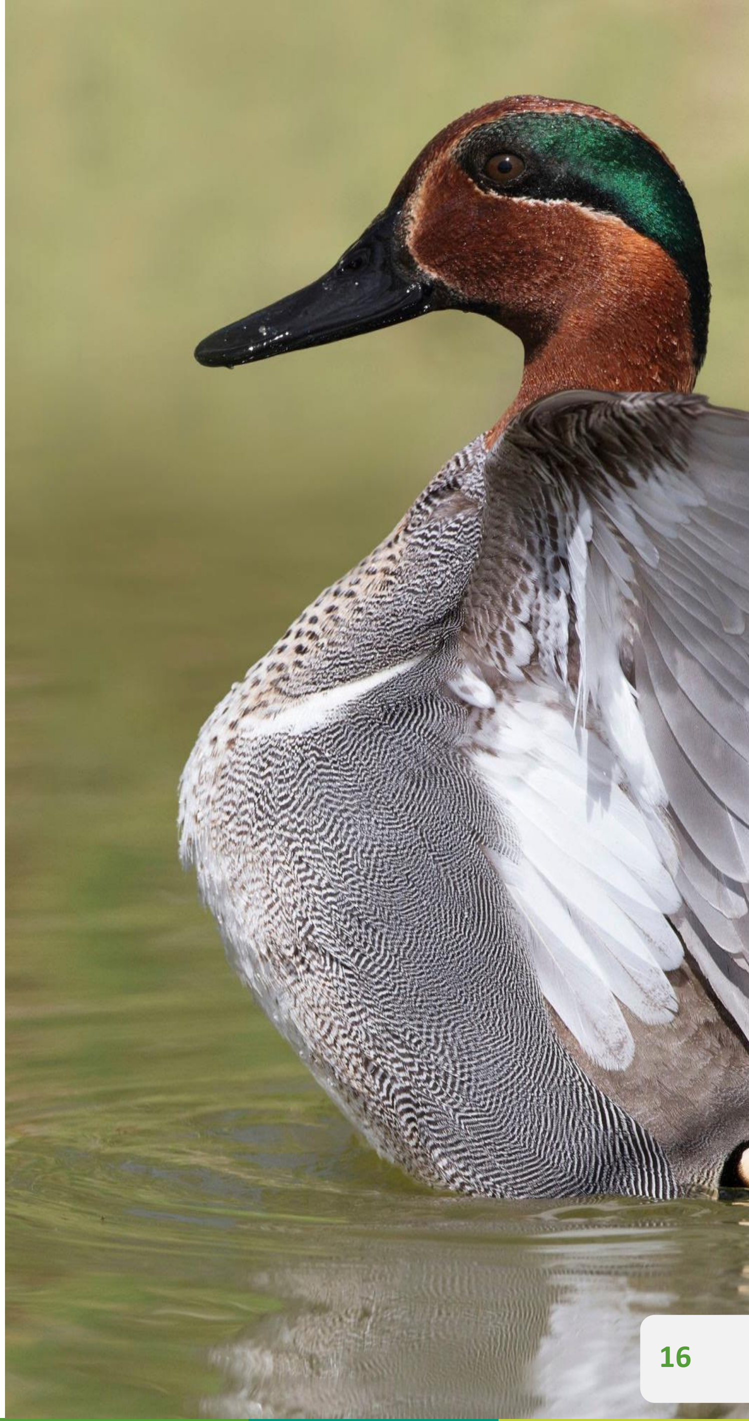


FIG 16-17. The pseudo-code of H-KASA & H-KASEO.

```
Initialize Keshtels population
Calculate the fitness and sort them in three types: N1, N2, and
N3
X*=the best solution.
while (t < maximum number of iteration)
    for each N1
        Do SA
            if the fitness of this new position is better than prior
                Update this lucky Keshtel.
            endif
    endfor
    for each N2
        Move the Keshtel between the two Keshtels.
    endfor
    for each N3
        Generate new Keshtels as new random solutions.
    endfor
    Update T;
    Merge the N1, N2, and N3
    Sort the Keshtels and form N1, N2, and N3 for next iteration
    Update the X* if there is better solution.
    t=t+1;
end while
return X*
```

```
Initialize Keshtels population
Calculate the fitness and sort them in three types: N1, N2, and
N3
X*=the best solution.
while (t < maximum number of iteration)
    for each N1
        Do SEO
            if the fitness of this new position is better than prior
                Update this lucky Keshtel.
            endif
    endfor
    for each N2
        Move the Keshtel between the two Keshtels.
    endfor
    for each N3
        Generate new Keshtels as new random solutions.
    endfor
    Update T;
    Merge the N1, N2, and N3
    Sort the Keshtels and form N1, N2, and N3 for next iteration
    Update the X* if there is better solution.
    t=t+1;
end while
return X*
```

Experimental Results

Table 7
Problem classification.

Classification	Instance	Problem size (f, r, s, gm, fm, dm, lm, pm, mm, cm, c, sm, P, T[month])
Small	SP1	(3,3,3,1,1,1,1,1,1,1,1,1,6,1)
	SP2	(10,10,10,3,3,3,3,3,3,3,5,3,6,3)
	SP3	(15,15,15,6,6,6,6,6,6,6,10,6,6,6)
Medium	MP4	(30,30,30,6,6,6,6,6,6,6,15,6,6,9)
	MP5	(40,45,45,8,8,8,8,8,8,8,20,8,6,11)
	MP6	(50,60,60,12,12,9,9,9,9,9,25,12,6,12)
	MP7	(50,60,60,15,15,10,10,10,10,10,25,20,6,12)
Large	LP8	(100,100,100,15,15,12,12,12,12,15,40,30,6,12)
	LP9	(150,150,150,25,25,20,20,20,20,20,60,30,6,12)
	LP10	(200,200,200,50,60,30,30,30,30,30,80,40,6,12)

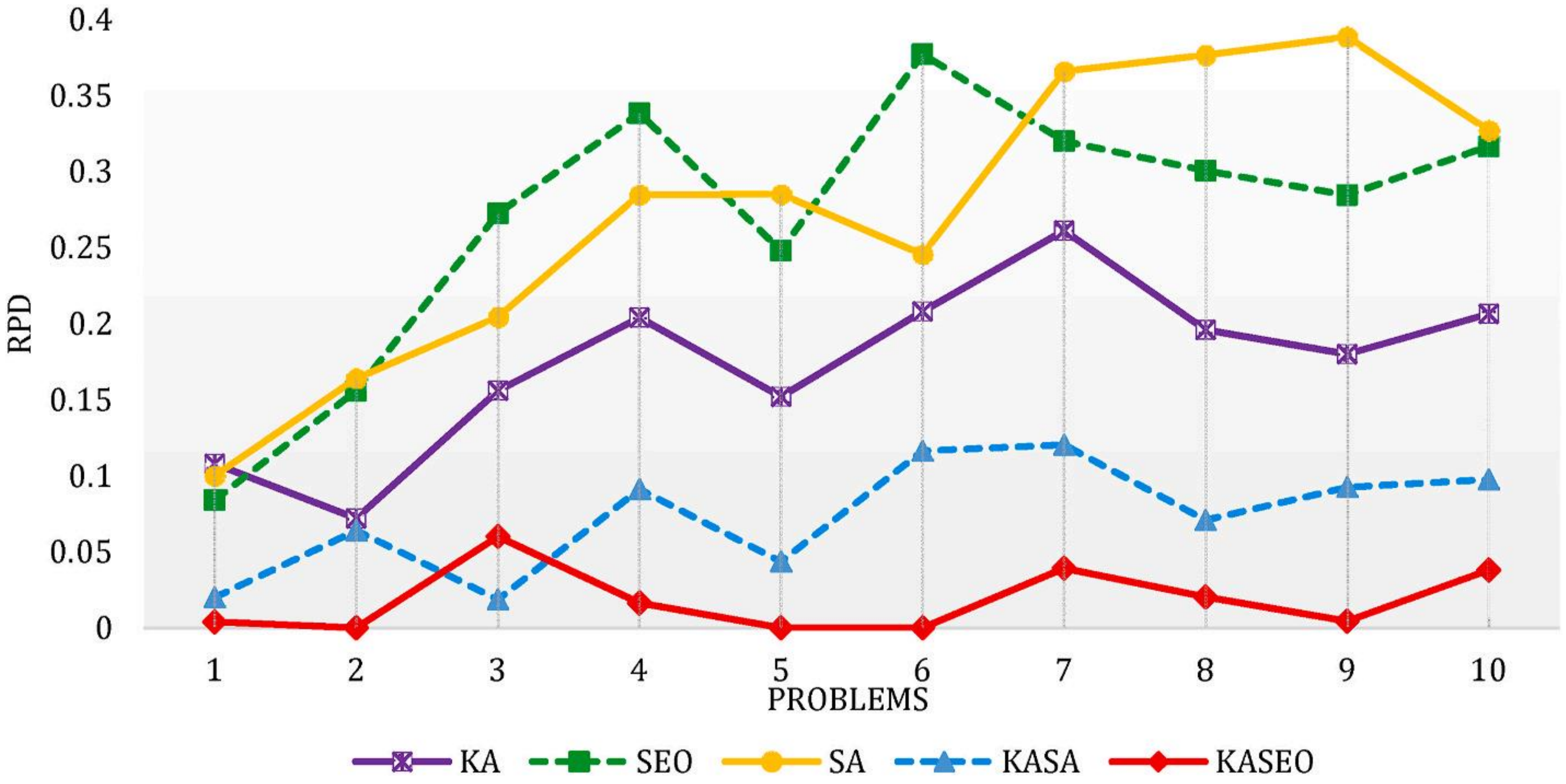
$$RPD = \frac{|Alg_{sol} - Min_{sol}|}{Min_{sol}}$$

Experimental Results

Table 11
The output results.

Objective	SP_1	SP_2	SP_3	MP_4	MP_5	MP_6	MP_7	LP_8	LP_9	LP_{10}
Gams	704.2	1016	2572	5037	7625	–	–	–	–	–
SA	785	1072	2843	5309	7460	9475	10910	13470	16238	19604
SEO	740	1056	2741	5307	7410	9474	10863	13243	16264	19550
KA	730	1048	2711	5294	7350	9268	10842	13095	16284	19490
KASA	718	1043	2638	5220	7310	9252	10633	13053	16199	19400
KASEO	705	1016	2618	5194	7270	9121	10529	13042	16151	19320

FIG 18. The mean RPD of each algorithm.



Q & A