A Parallel Novelty Search Metaheuristic Applied to a Wildfire Prediction System

12th IEEE Workshop Parallel / Distributed Combinatorics and Optimization (PDCO 2022)

Jan Strappa[ab], Paola Caymes Scutari[ab] and Germán Bianchini[a]

[a] Laboratory of Research in Distributed/Parallel Computing (LICPaD),
Systems Engineering Dept. - National Technological University - Mendoza - Argentina.
[b] National Scientific and Technical Research Council (CONICET)

June 3, 2022

The wildfire propagation problem

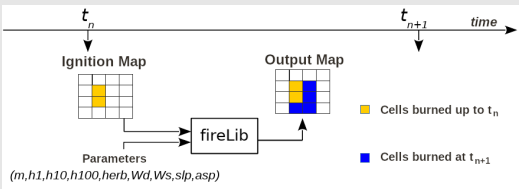
Wildfires

- Uncontrolled fires of rapid spread that affect land with vegetation such as forests, plains, grasslands, pastures, among others.
- Loss of human lives, evacuations, damage to flora and fauna, atmospheric emissions, soil erosion, major economic impacts.
- Millions of hectares per year burn globally; fire disasters more likely every year

Wilfdire in Chubut, Argentina (March, 2021).

Prediction problem

Fire Simulator

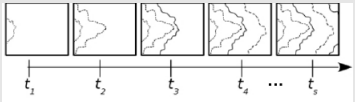


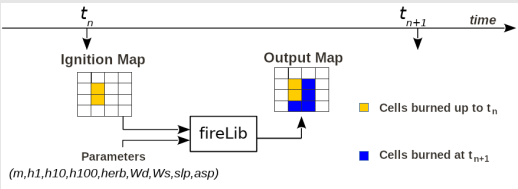
Input: scenarios (parameter vectors with environmental information)

Output: labeled map

Prediction problem

Fire Simulator

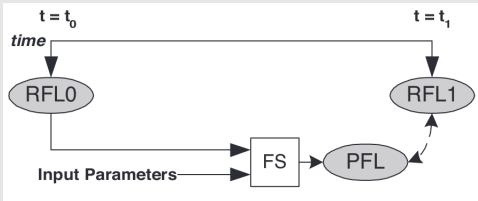




Input: scenarios (parameter vectors with environmental information)

Output: labeled map

Classic Prediction



RFL0, RFL1: real fire line at times t_0, t_1
PFL: predicted fire line
Input parameters: scenario
FS: fire simulator

Uncertainty

Parameter	Description	Range	Unit of measurement
Model	Rothermel Fuel Model	1-13	fuel model
WindSpd	Wind speed	0-80	miles/hour
WindDir	Wind direction	0-360	degrees clockwise from North
M1	Dead Fuel Moisture in 1 hour since start of fire	1-60	percent
M10	Dead Fuel Moisture in 10 h	1-60	percent
M100	Dead Fuel Moisture in 100 h	1-60	percent
Mherb	Live herbaceous fuel moisture	30-300	percent
Slope	Surface slope	0-81	degrees
Aspect	Direction of the surface faces	0-360	degrees clockwise from north

Sources of uncertainty in input parameters

Uncertainty

Parameter	Description	Range	Unit of measurement
Model	Rothermel Fuel Model	1-13	fuel model
WindSpd	Wind speed	0-80	miles/hour
WindDir	Wind direction	0-360	degrees clockwise from North
M1	Dead Fuel Moisture in 1 hour since start of fire	1-60	percent
M10	Dead Fuel Moisture in 10 h	1-60	percent
M100	Dead Fuel Moisture in 100 h	1-60	percent
Mherb	Live herbaceous fuel moisture	30-300	percent
Slope	Surface slope	0-81	degrees
Aspect	Direction of the surface faces	0-360	degrees clockwise from north

Sources of uncertainty in input parameters

- imprecise measurements

Uncertainty

Parameter	Description	Range	Unit of measurement
Model	Rothermel Fuel Model	1-13	fuel model
WindSpd	Wind speed	0-80	miles/hour
WindDir	Wind direction	0-360	degrees clockwise from North
M1	Dead Fuel Moisture in 1 hour since start of fire	1-60	percent
M10	Dead Fuel Moisture in 10 h	1-60	percent
M100	Dead Fuel Moisture in 100 h	1-60	percent
Mherb	Live herbaceous fuel moisture	30-300	percent
Slope	Surface slope	0-81	degrees
Aspect	Direction of the surface faces	0-360	degrees clockwise from north

Sources of uncertainty in input parameters

- imprecise measurements
- dynamically changing variables

Uncertainty

Parameter	Description	Range	Unit of measurement
Model	Rothermel Fuel Model	1-13	fuel model
WindSpd	Wind speed	0-80	miles/hour
WindDir	Wind direction	0-360	degrees clockwise from North
M1	Dead Fuel Moisture in 1 hour since start of fire	1-60	percent
M10	Dead Fuel Moisture in 10 h	1-60	percent
M100	Dead Fuel Moisture in 100 h	1-60	percent
Mherb	Live herbaceous fuel moisture	30-300	percent
Slope	Surface slope	0-81	degrees
Aspect	Direction of the surface faces	0-360	degrees clockwise from north

Sources of uncertainty in input parameters

- imprecise measurements
- dynamically changing variables
- unknown values

Prediction methods and uncertainty reduction

- Classical prediction: 1 scenario/simulation, 1 result for prediction

Prediction methods and uncertainty reduction

- Classical prediction: 1 scenario/simulation, 1 result for prediction

Uncertainty Reduction Methods

- *Data-driven methods*: *n* scenarios/simulations, result obtained from 1 simulation

Prediction methods and uncertainty reduction

- Classical prediction: 1 scenario/simulation, 1 result for prediction

Uncertainty Reduction Methods

- *Data-driven methods*: *n* scenarios/simulations, result obtained from 1 simulation
- ***DDM with Multiple Overlapping Solutions (DDM-MOS)***: *n* scenarios/simulations, results obtained from subset of the *n* simulations

Prediction methods and uncertainty reduction

- Classical prediction: 1 scenario/simulation, 1 result for prediction

Uncertainty Reduction Methods

- *Data-driven methods*: *n* scenarios/simulations, result obtained from 1 simulation
- ***DDM with Multiple Overlapping Solutions (DDM-MOS)***: *n* scenarios/simulations, results obtained from subset of the *n* simulations
  - **ESS (*Evolutionary Statistical System*)**: optimization with metaheuristics, M/W

Prediction methods and uncertainty reduction

- Classical prediction: 1 scenario/simulation, 1 result for prediction

Uncertainty Reduction Methods

- *Data-driven methods*: *n* scenarios/simulations, result obtained from 1 simulation
- ***DDM with Multiple Overlapping Solutions (DDM-MOS)***: *n* scenarios/simulations, results obtained from subset of the *n* simulations
  - **ESS (*Evolutionary Statistical System*)**: optimization with metaheuristics, M/W
    - **ESSIM (*Evolutionary Statistical System with Island Model*):**
      double M/W hierarchy (Monitor → Masters of Islands → Workers).
      → **ESSIM-EA**: evolutionary algorithm (GA)
      → **ESSIM-DE**: differential evolution

Predecessors: ESS

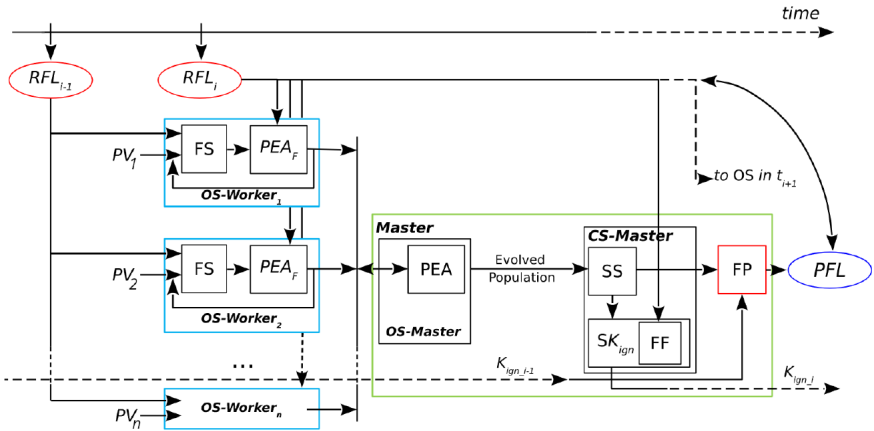


Fig. 1. Evolutionary Statistical System. FS: fire simulator, *OS-Master*: Optimization Stage in Master, *OS-Worker*$_x$: Optimization Stage in Worker x, PEA: Parallel Evolutionary Algorithms, PEA_F: Parallel Evolutionary Algorithm (fitness evaluation), *CS-Master*: Calibration Stage in Master, FF: fitness function, RFL_i: real fire line of instant t_i, PFL_i: predicted fire line of instant t_i, $PV_{\{1...n\}}$: parameter vectors (scenarios), t_n: time instant n, K_{ign}: *Key Ignition Value* for t_i, FP: Fire Prediction stage, SS: Statistical Stage, SK_{ign}: *Key Ignition Value* search.

PEA$_F$ computes the fitness $f(x)$ by comparing simulated vs real map (with Jaccard's index)

Previous methods: ESSIM-EA y ESSIM-DE

Main contributions of existing systems

- ESSIM-EA: improved quality and execution times than its predecessors
- ESSIM-DE: shorter execution times than ESSIM-EA, reaches acceptable quality fast but converges prematurely

Previous methods: ESSIM-EA y ESSIM-DE

Main contributions of existing systems

- ESSIM-EA: improved quality and execution times than its predecessors
- ESSIM-DE: shorter execution times than ESSIM-EA, reaches acceptable quality fast but converges prematurely

Limitations

- ESSIM-EA uses solutions from the last evolved population: good solutions may be lost in previous generations.
- ESSIM-DE presents premature convergence, quality improvements only with dynamic tuning techniques for diversifying results through randomness.

Previous methods: ESSIM-EA y ESSIM-DE

Main contributions of existing systems

- ESSIM-EA: improved quality and execution times than its predecessors
- ESSIM-DE: shorter execution times than ESSIM-EA, reaches acceptable quality fast but converges prematurely

Limitations

- ESSIM-EA uses solutions from the last evolved population: good solutions may be lost in previous generations.
- ESSIM-DE presents premature convergence, quality improvements only with dynamic tuning techniques for diversifying results through randomness.

A proposal that may overcome these limitations is to use the **Novelty Search** paradigm.

Novelty Search

Lehman, Joel & Stanley, Kenneth. (2008). Exploiting open-endedness to solve problems through the search for novelty. Artificial Life – ALIFE.

Motivation

In highly complex search problems, the **objective function** (in this case, the *fitness function*) is not enough to reach a solution, and can even prevent the search from doing so. Consequences:

- premature convergence
- local optima

Novelty-based paradigm

Guide the search by the **novelty** of solutions instead of guiding it by the **objective function**.

Novelty Search approach

Description

- Define a **behavioral characterization** for the individuals (e.g., *fitness difference*).

genotype $x = (m, h1, h10, h100, herb, Wd, Ws, slp, asp)$	phenotype *map simulated from x*	fitness $f(x)$
$(1, 40, 30, 10, 57, 45, 20, 65, 18)$		**0.87**

Novelty Search approach

Description

- Define a **behavioral characterization** for the individuals (e.g., *fitness difference*).

genotype $x = (m, h1, h10, h100, herb, Wd, Ws, slp, asp)$	phenotype *map simulated from x*	fitness $f(x)$
$(1, 40, 30, 10, 57, 45, 20, 65, 18)$		**0.87**

- A **novelty score** function is computed for each new individual based on its behavior (w.r.t. others).

Novelty Search approach

Description

- Define a **behavioral characterization** for the individuals (e.g., *fitness difference*).

genotype	phenotype	fitness
$x = (m, h1, h10, h100, herb, Wd, Ws, slp, asp)$	*map simulated from x*	$f(x)$
$(1, 40, 30, 10, 57, 45, 20, 65, 18)$		**0.87**

- A **novelty score** function is computed for each new individual based on its behavior (w.r.t. others).
- The most novel individuals are selected:

Novelty Search approach

Description

- Define a **behavioral characterization** for the individuals (e.g., *fitness difference*).

genotype	phenotype	fitness
$x = (m, h1, h10, h100, herb, Wd, Ws, slp, asp)$	*map simulated from x*	$f(x)$
$(1, 40, 30, 10, 57, 45, 20, 65, 18)$		**0.87**

- A **novelty score** function is computed for each new individual based on its behavior (w.r.t. others).
- The most novel individuals are selected:
  - for **population replacement**;

Novelty Search approach

Description

- Define a **behavioral characterization** for the individuals (e.g., *fitness difference*).

genotype	phenotype	fitness
$x = (m, h1, h10, h100, herb, Wd, Ws, slp, asp)$	*map simulated from x*	$f(x)$
$(1, 40, 30, 10, 57, 45, 20, 65, 18)$		**0.87**

- A **novelty score** function is computed for each new individual based on its behavior (w.r.t. others).
- The most novel individuals are selected:
  - for **population replacement**;
  - for inclusion in an **archive of past novel solutions**.

Novelty Search approach

Description

- Define a **behavioral characterization** for the individuals (e.g., *fitness difference*).

genotype $x = (m, h1, h10, h100, herb, Wd, Ws, slp, asp)$	phenotype *map simulated from x*	fitness $f(x)$
$(1, 40, 30, 10, 57, 45, 20, 65, 18)$		**0.87**

- A **novelty score** function is computed for each new individual based on its behavior (w.r.t. others).
- The most novel individuals are selected:
  - for **population replacement**;
  - for inclusion in an **archive of past novel solutions**.
- The solution(s) of highest fitness are registered during the search.

Novelty Search approach

Novelty score function (*sparseness*)

The novelty of an individual is computed w.r.t. other individuals from the **current population** and the **archive of past novel individuals**.

$$\rho(x) = \frac{1}{k} \sum_{i=0}^{k-1} dist(x, \mu_i)$$

dist: measure of behavioral difference between two individuals in the search space.
k: number of closest neighbors (w.r.t. *dist*) to consider in the score.
μ_i: i-th closest neighbor of x.

Advantages of using *Novelty Search*

Advantages

- Avoids *by design* issues such as premature convergence (maximizes exploration).
- Easy implementation by modifying an existing metaheuristic:
  - replacing objective function by a measure of novelty;
  - adding an archive.

Proposal: Framework for *ESS - Novelty Search*

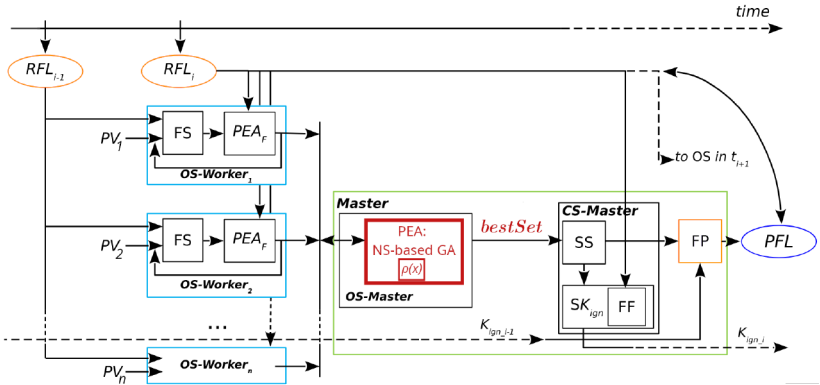


Fig. 3. Evolutionary Statistical System - Novelty Search. FS: fire simulator, *OS-Master*: Optimization Stage in Master, *OS-Worker*$_x$: Optimization Stage in Worker x, PEA: Parallel Evolutionary Algorithm, *NS-based GA*: Novelty Search-based Genetic Algorithm, $\rho(x)$: novelty score function (1), PEA_F: Parallel Evolutionary Algorithm (fitness evaluation), *CS-Master*: Calibration Stage in Master, FF: fitness function, RFL_i: real fire line of instant t_i, PFL_i: predicted fire line of instant t_i, $PV_{\{1...n\}}$: parameter vectors (scenarios), t_n: time instant n, K_{ign}: *Key Ignition Value* for t_i, FP: Fire Prediction stage, SS: Statistical Stage, SK_{ign}: *Key Ignition Value* search.

$$\rho(x) = \frac{1}{k}\sum_{i=0}^{k-1} f(x) - f(\mu_i) \qquad (f(x) \text{ is computed at } PEA_F \text{ by workers})$$

Conclusions and future work

Conclusions

- Preliminary results suggest NS obtained reasonable quality (in one case comparable to the best methods), and execution times comparable to the most efficient method.
- Promising results with straight-forward implementation, with many possibilities for improvement.
- This approach can be applied and adapted to other kinds of propagation problem.

Conclusions and future work

Conclusions

- Preliminary results suggest NS obtained reasonable quality (in one case comparable to the best methods), and execution times comparable to the most efficient method.
- Promising results with straight-forward implementation, with many possibilities for improvement.
- This approach can be applied and adapted to other kinds of propagation problem.

Future work

- Use a different behavior characterization, e.g., simulated map.
- Improvements based on parameter calibration (k, archive & result set size), underlying metaheuristic, management of archive, etc.
- Algorithm variants with different criteria for the aggregation of results (not just fitness).
- Explore: parallelization, e.g., with an island model; hybridization with fitness-based search.

A Parallel Novelty Search Metaheuristic Applied to a Wildfire Prediction System

Thank you!

Questions?

jstrappa@frm.utn.edu.ar

Additional slides

```
1: population ⟵ initializePopulation(N)
2: archive ⟵ ∅
3: bestSet ⟵ ∅
4: generations ⟵ 0
5: maxFitness ⟵ 0
6: while generations < maxGen and maxFitness < fThreshold do
7:     offspring ⟵ generateOffspring(population, m, mR, cR)
8:     for each individual ind ∈ (population ∪ offspring) do
9:         ind. fitness ⟵ evaluateFitness(ind)
10:    end for
11:    noveltySet ⟵ (population ∪ offspring ∪ archive)
12:    for each individual ind ∈ (population ∪ offspring) do
13:        ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:    end for
15:    archive ⟵ updateArchive(archive, offspring)
16:    population ⟵ replaceByNovelty(population, offspring, N)
17:    bestSet ⟵ updateBest(bestSet, offspring)
18:    maxFitness ⟵ getMaxFitness(bestSet)
19:    generations ⟵ generations + 1
20: end while
21: return bestSet
```

INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
$maxGen$	max # of generations
$fThreshold$	fitness threshold
OUTPUT	
$bestSet$	solutions

```
1: population ⟵ initializePopulation(N)
2: archive ⟵ ∅
3: bestSet ⟵ ∅
4: generations ⟵ 0
5: maxFitness ⟵ 0
6: while generations < maxGen and maxFitness < fThreshold do
7:     offspring ⟵ generateOffspring(population, m, mR, cR)
8:     for each individual ind ∈ (population ∪ offspring) do
9:         ind. fitness ⟵ evaluateFitness(ind)
10:    end for
11:    noveltySet ⟵ (population ∪ offspring ∪ archive)
12:    for each individual ind ∈ (population ∪ offspring) do
13:        ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:    end for
15:    archive ⟵ updateArchive(archive, offspring)
16:    population ⟵ replaceByNovelty(population, offspring, N)
17:    bestSet ⟵ updateBest(bestSet, offspring)
18:    maxFitness ⟵ getMaxFitness(bestSet)
19:    generations ⟵ generations + 1
20: end while
21: return bestSet
```

INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
$maxGen$	max # of generations
$fThreshold$	fitness threshold
OUTPUT	
$bestSet$	solutions

```
1:  population ⟵ initializePopulation(N)
2:  archive ⟵ ∅
3:  bestSet ⟵ ∅
4:  generations ⟵ 0
5:  maxFitness ⟵ 0
6:  while generations < maxGen and maxFitness < fThreshold do
7:      offspring ⟵ generateOffspring(population, m, mR, cR)
8:      for each individual ind ∈ (population ∪ offspring) do
9:          ind. fitness ⟵ evaluateFitness(ind)
10:     end for
11:     noveltySet ⟵ (population ∪ offspring ∪ archive)
12:     for each individual ind ∈ (population ∪ offspring) do
13:         ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:     end for
15:     archive ⟵ updateArchive(archive, offspring)
16:     population ⟵ replaceByNovelty(population, offspring, N)
17:     bestSet ⟵ updateBest(bestSet, offspring)
18:     maxFitness ⟵ getMaxFitness(bestSet)
19:     generations ⟵ generations + 1
20: end while
21: return bestSet
```

INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
$maxGen$	max # of generations
$fThreshold$	fitness threshold
OUTPUT	
$bestSet$	solutions

```
1: population ⟵ initializePopulation(N)
2: archive ⟵ ∅
3: bestSet ⟵ ∅
4: generations ⟵ 0
5: maxFitness ⟵ 0
6: while generations < maxGen and maxFitness < fThreshold do
7:     offspring ⟵ generateOffspring(population, m, mR, cR)
8:     for each individual ind ∈ (population ∪ offspring) do
9:         ind. fitness ⟵ evaluateFitness(ind)
10:    end for
11:    noveltySet ⟵ (population ∪ offspring ∪ archive)
12:    for each individual ind ∈ (population ∪ offspring) do
13:        ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:    end for
15:    archive ⟵ updateArchive(archive, offspring)
16:    population ⟵ replaceByNovelty(population, offspring, N)
17:    bestSet ⟵ updateBest(bestSet, offspring)
18:    maxFitness ⟵ getMaxFitness(bestSet)
19:    generations ⟵ generations + 1
20: end while
21: return bestSet
```

INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
$maxGen$	max # of generations
$fThreshold$	fitness threshold
OUTPUT	
$bestSet$	solutions

```
1: population ⟵ initializePopulation(N)
2: archive ⟵ ∅
3: bestSet ⟵ ∅
4: generations ⟵ 0
5: maxFitness ⟵ 0
6: while generations < maxGen and maxFitness < fThreshold do
7:     offspring ⟵ generateOffspring(population, m, mR, cR)
8:     for each individual ind ∈ (population ∪ offspring) do
9:         ind. fitness ⟵ evaluateFitness(ind)
10:    end for
11:    noveltySet ⟵ (population ∪ offspring ∪ archive)
12:    for each individual ind ∈ (population ∪ offspring) do
13:        ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:    end for
15:    archive ⟵ updateArchive(archive, offspring)
16:    population ⟵ replaceByNovelty(population, offspring, N)
17:    bestSet ⟵ updateBest(bestSet, offspring)
18:    maxFitness ⟵ getMaxFitness(bestSet)
19:    generations ⟵ generations + 1
20: end while
21: return bestSet
```

$$\rho(x) = \frac{1}{k}\sum_{i=0}^{k-1} dist(x, \mu_i)$$

INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
maxGen	max # of generations
fThreshold	fitness threshold
OUTPUT	
bestSet	solutions

```
1: population ⟵ initializePopulation(N)
2: archive ⟵ ∅
3: bestSet ⟵ ∅
4: generations ⟵ 0
5: maxFitness ⟵ 0
6: while generations < maxGen and maxFitness < fThreshold do
7:     offspring ⟵ generateOffspring(population, m, mR, cR)
8:     for each individual ind ∈ (population ∪ offspring) do
9:         ind. fitness ⟵ evaluateFitness(ind)
10:    end for
11:    noveltySet ⟵ (population ∪ offspring ∪ archive)
12:    for each individual ind ∈ (population ∪ offspring) do
13:        ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:    end for
15:    archive ⟵ updateArchive(archive, offspring)
16:    population ⟵ replaceByNovelty(population, offspring, N)
17:    bestSet ⟵ updateBest(bestSet, offspring)
18:    maxFitness ⟵ getMaxFitness(bestSet)
19:    generations ⟵ generations + 1
20: end while
21: return bestSet
```

INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
$maxGen$	max # of generations
$fThreshold$	fitness threshold
OUTPUT	
$bestSet$	solutions

```
1: population ⟵ initializePopulation(N)
2: archive ⟵ ∅
3: bestSet ⟵ ∅
4: generations ⟵ 0
5: maxFitness ⟵ 0
6: while generations < maxGen and maxFitness < fThreshold do
7:     offspring ⟵ generateOffspring(population, m, mR, cR)
8:     for each individual ind ∈ (population ∪ offspring) do
9:         ind. fitness ⟵ evaluateFitness(ind)
10:    end for
11:    noveltySet ⟵ (population ∪ offspring ∪ archive)
12:    for each individual ind ∈ (population ∪ offspring) do
13:        ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:    end for
15:    archive ⟵ updateArchive(archive, offspring)
16:    population ⟵ replaceByNovelty(population, offspring, N)
17:    bestSet ⟵ updateBest(bestSet, offspring)
18:    maxFitness ⟵ getMaxFitness(bestSet)
19:    generations ⟵ generations + 1
20: end while
21: return bestSet
```

INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
$maxGen$	max # of generations
$fThreshold$	fitness threshold
OUTPUT	
$bestSet$	solutions

```
1: population ⟵ initializePopulation(N)
2: archive ⟵ ∅
3: bestSet ⟵ ∅
4: generations ⟵ 0
5: maxFitness ⟵ 0
6: while generations < maxGen and maxFitness < fThreshold do
7:     offspring ⟵ generateOffspring(population, m, mR, cR)
8:     for each individual ind ∈ (population ∪ offspring) do
9:         ind. fitness ⟵ evaluateFitness(ind)
10:    end for
11:    noveltySet ⟵ (population ∪ offspring ∪ archive)
12:    for each individual ind ∈ (population ∪ offspring) do
13:        ind.novelty ⟵ evaluateNovelty(ind, noveltySet, k)
14:    end for
15:    archive ⟵ updateArchive(archive, offspring)
16:    population ⟵ replaceByNovelty(population, offspring, N)
17:    bestSet ⟵ updateBest(bestSet, offspring)
18:    maxFitness ⟵ getMaxFitness(bestSet)
19:    generations ⟵ generations + 1
20: end while
21: return bestSet
```

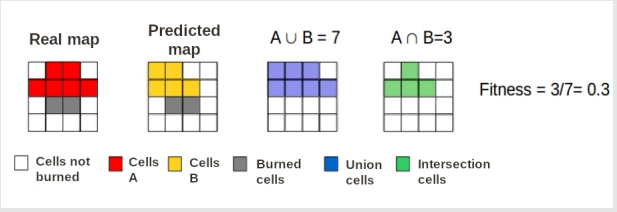
INPUT	
N	population size
m	# of descendants
mR	mutation rate
r	tournament prob.
k	neighbors (for $\rho(x)$)
maxGen	max # of generations
fThreshold	fitness threshold
OUTPUT	
bestSet	solutions

Predecessors: ESS, ESSIM-EA and ESSIM-DE

Optimization

Fitness function computation for each simulation uses **Jaccard's Index**:

$$fitness(A, B) = \frac{A \cap B}{A \cup B}$$



Predecessors: ESSIM-EA and ESSIM-DE