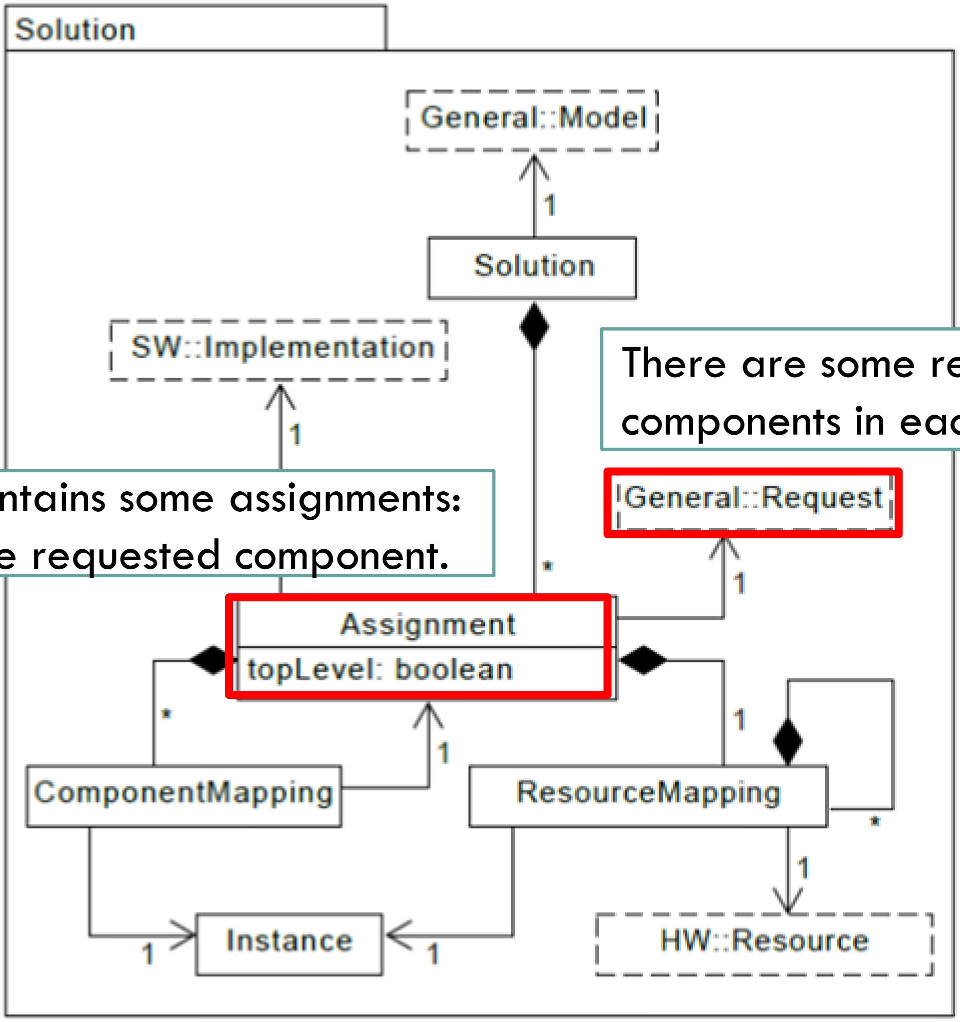
SOLVING THE QUALITY-BASED SOFTWARE-SELECTION AND HARDWARE-MAPPING PROBLEM WITH ACO

Model-Driven Software Engineering Research Group
Department of Software Engineering
University of Isfahan

Samaneh Hoseindoost s.hoseindoost@eng.ui.ac.ir
Meysam Karimi meysam.karimi@eng.ui.ac.ir
Shekoufeh Kolahdouz-Rahimi sh.rahimi@eng.ui.ac.ir
Bahman Zamani zamani@eng.ui.ac.ir

SOLUTION OVERVIEW



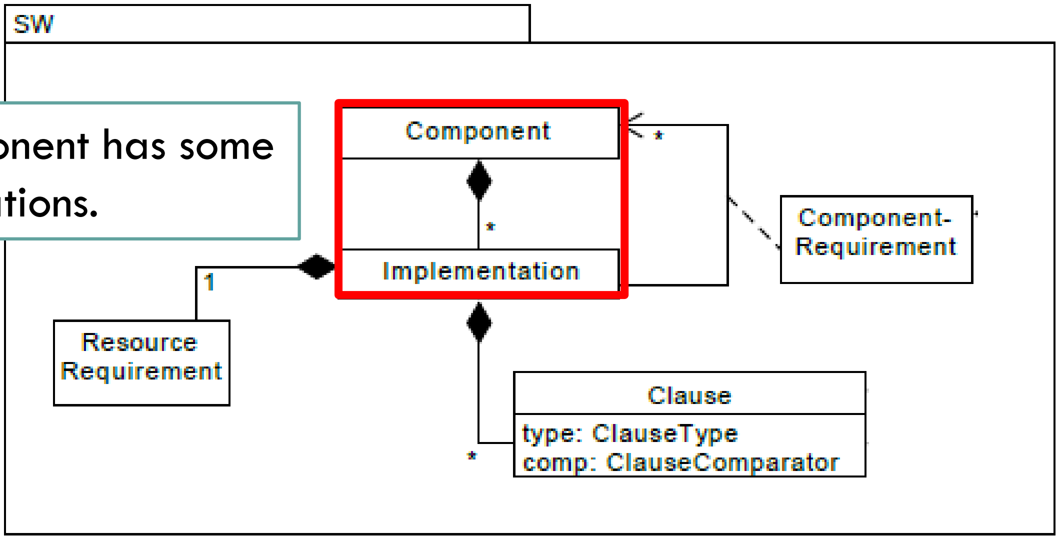
There are some requested components in each benchmark.

The solution model should contains some assignments:

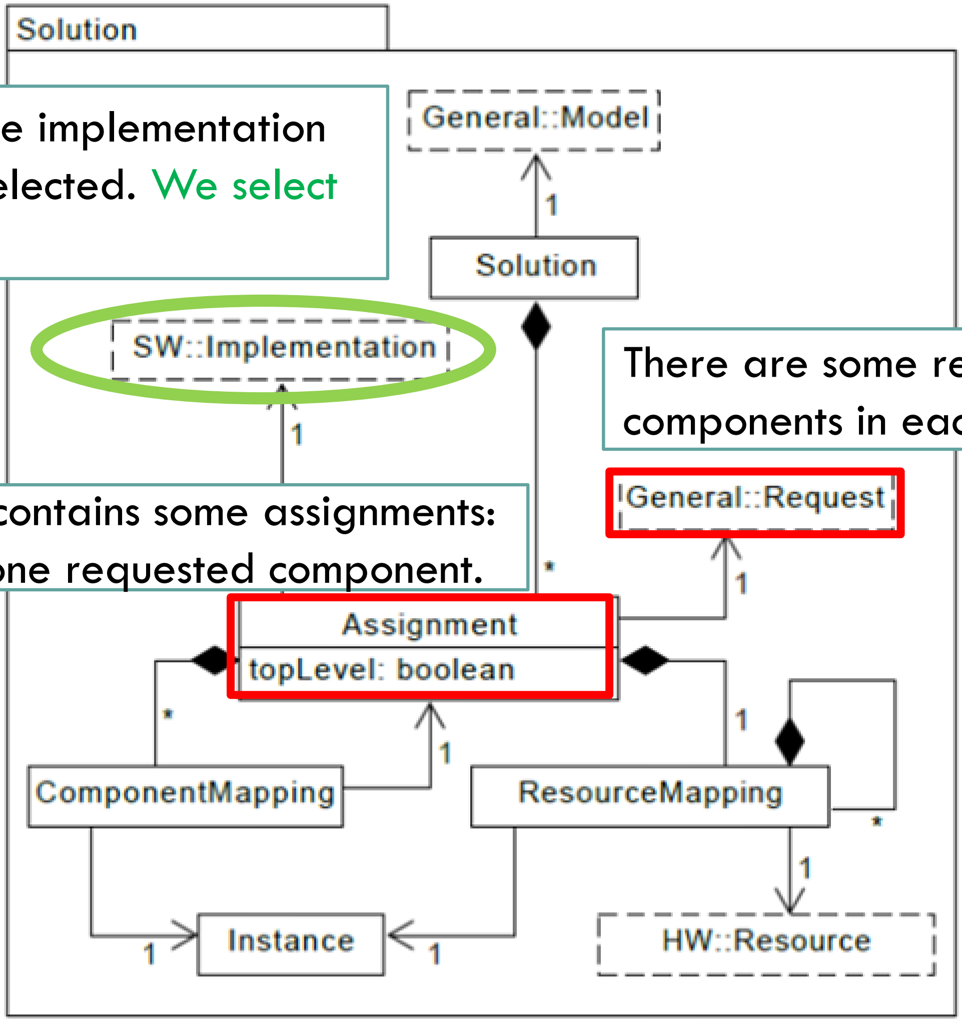
- one assignment for one requested component.

PARTIAL OF SOFTWARE MODEL

Each component has some implementations.



SOLUTION OVERVIEW



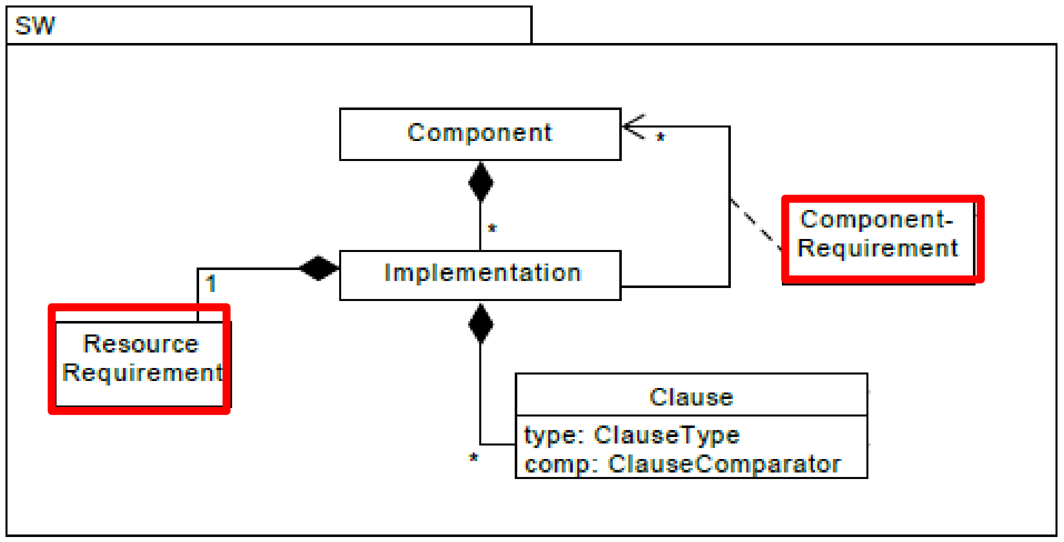
For each assignment, one of the implementation of the component, should be selected. We select it randomly.

There are some requested components in each benchmark.

The solution model should contains some assignments:

- one assignment for one requested component.

PARTIAL OF SOFTWARE MODEL



Each implementation has a resource requirement and some component requirements.

SOLUTION OVERVIEW

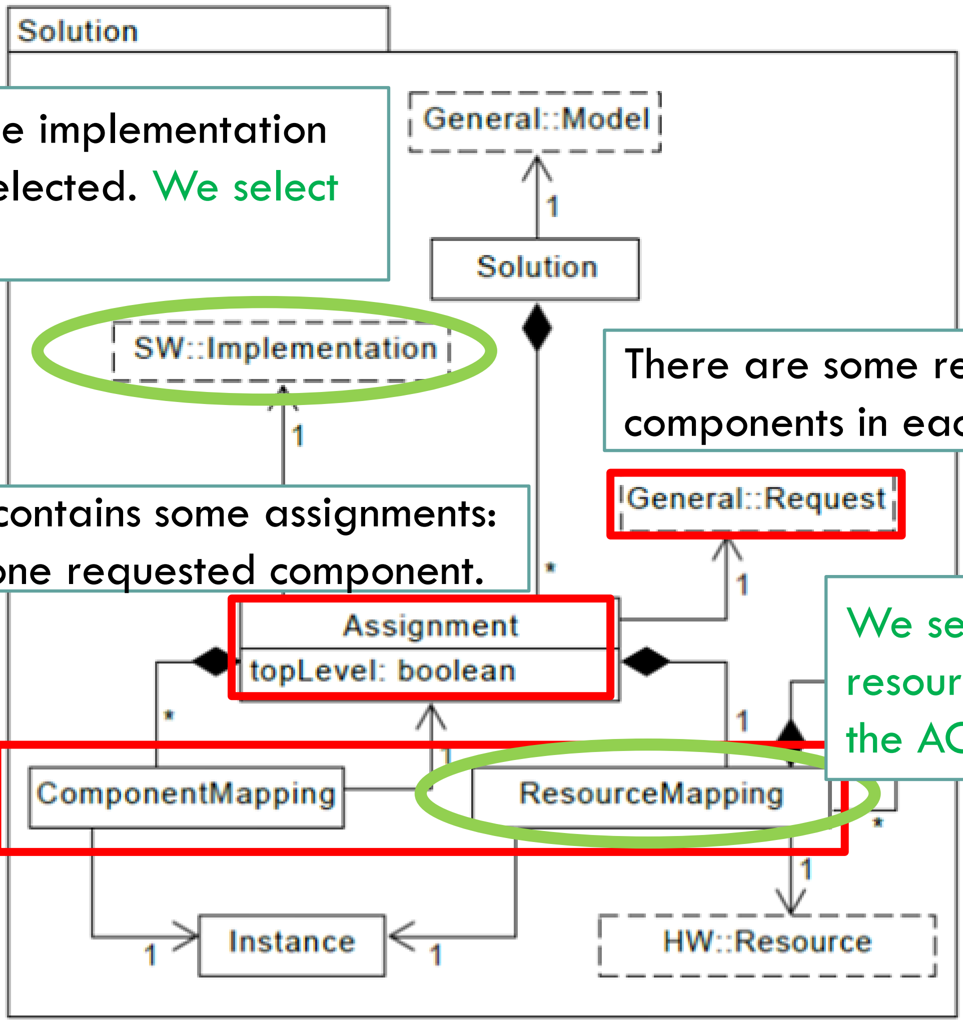
For each assignment, one of the implementation of the component, should be selected. We select it randomly.

There are some requested components in each benchmark.

The solution model should contains some assignments:

- one assignment for one requested component.

We select suitable resources according to the ACO algorithm.



THE GOAL:

- The constraint is that:
  - At most one implementation is deployed on each resource. So a resource could not be used in two assignments.

- For each assignment we have to find suitable component and resource mappings such that, all required property clauses of the implementation and all request constraints are fulfilled.

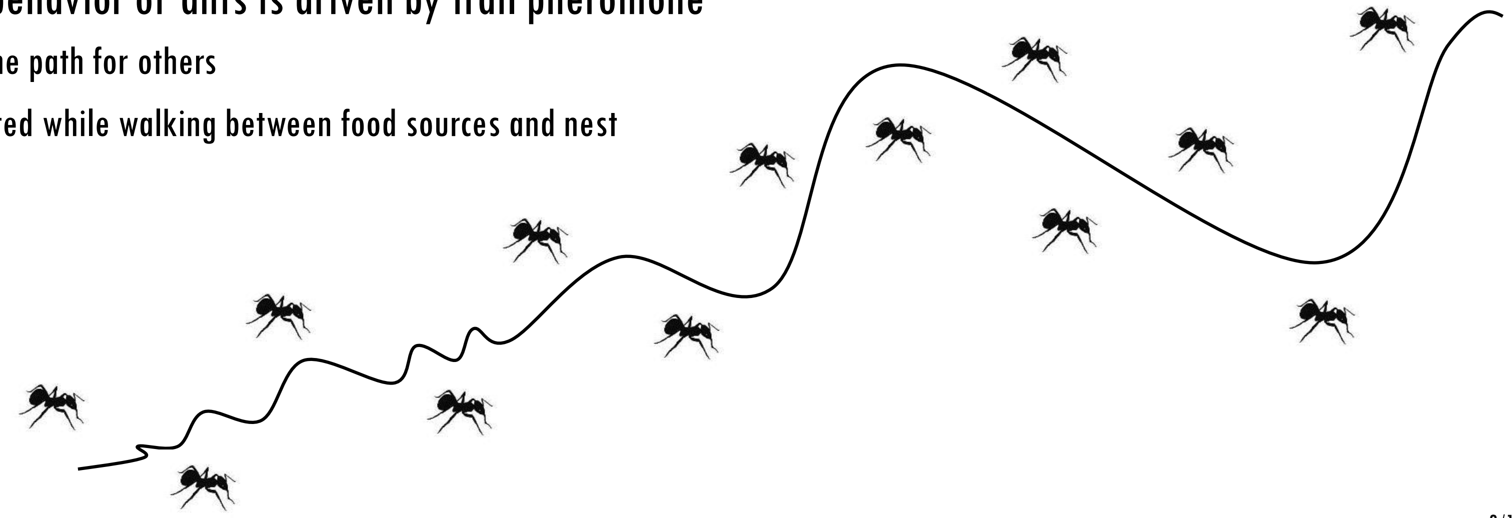
- The final goal is to achieve a valid solution with minimum objective that is constructed in minimum time.

ACO ALGORITHM OVERVIEW

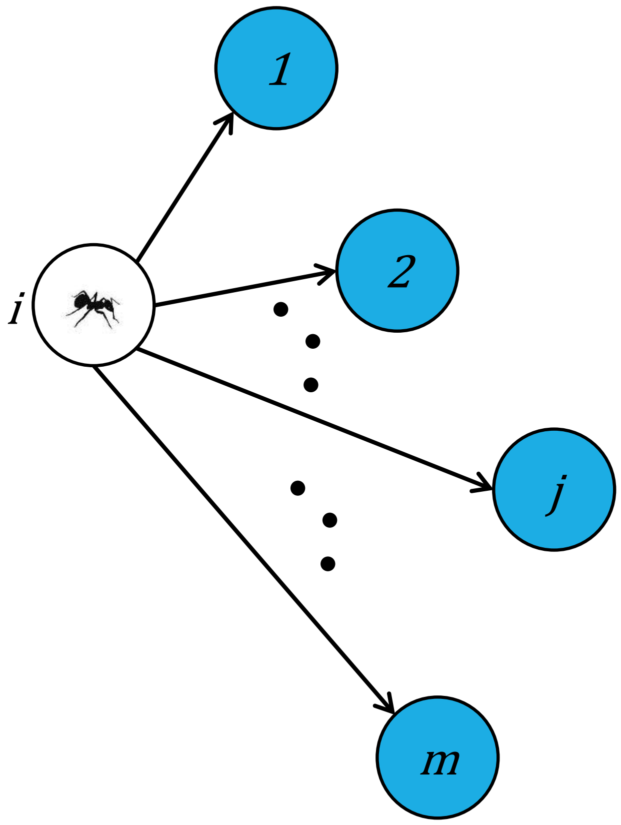
- "Ant Colony Optimization (ACO) studies artificial systems that take inspiration from the behavior of real ant colonies and which are used to solve discrete optimization problems."

Source: ACO website, http://iridia.ulb.ac.be/~mdorigo/ACO/about.html

- Real ants use stigmergy. How?
  - PHEROMONES!!!
- Social behavior of ants is driven by trail pheromone
  - Mark the path for others
  - Deposited while walking between food sources and nest



ACO ALGORITHM OVERVIEW



Probability of path selection:

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{\forall j}(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$$

Deposit Pheromone and Evaporation:

$$\tau_{ij} = \tau_{ij} + \frac{Q}{objective\ of\ node\ ij}$$

$$\tau_{ij} = (1-\rho) * \tau_{ij}$$

Heuristic Information:

$$\eta_{ij} = \frac{1}{objective\ of\ node\ ij}$$

ACO SOLVER OVERVIEW

- ACO Solver has two main parts:
  - In part 1 valid software solutions are generated.
  - in part 2, the solutions are passed to swarm of ants for finding the best valid solutions.
- The population size and iteration size are predefined parameters that determine the termination of parts 1 and 2, respectively.

https://github.com/Ariyanic/TTC18/tree/master/jastadd-mquat-solver-aco

PART 1: GENERATING VALID SOFTWARE SOLUTIONS

```
Initialize population size;
Initialize $\tau_0, \alpha, \beta$;
For(int pop=0; i < population size; pop++){
    If time exceeds from maxSolvingTime: break;
    create an empty solution.
    For each requested components, create an empty assignment.
    For each assignment i{
            create a valid component mappings.
            find the possible resources (i.e. the resources that not violated the required property clauses and request constraints) from all of the available resources.
            If an assignment has no possible resources: Ignore the solution.
            Else: set $\tau_{ij} = \tau_0$, $\eta_{ij} = \frac{1}{assignment.objective}$ and $P_{ij} = \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{\forall j} (\tau_{ij})^\alpha (\eta_{ij})^\beta}$ for each possible resources j
    }
    Save the solutions that all  component mappings are valid and all assignments of it has at least one possible resource.
}
```

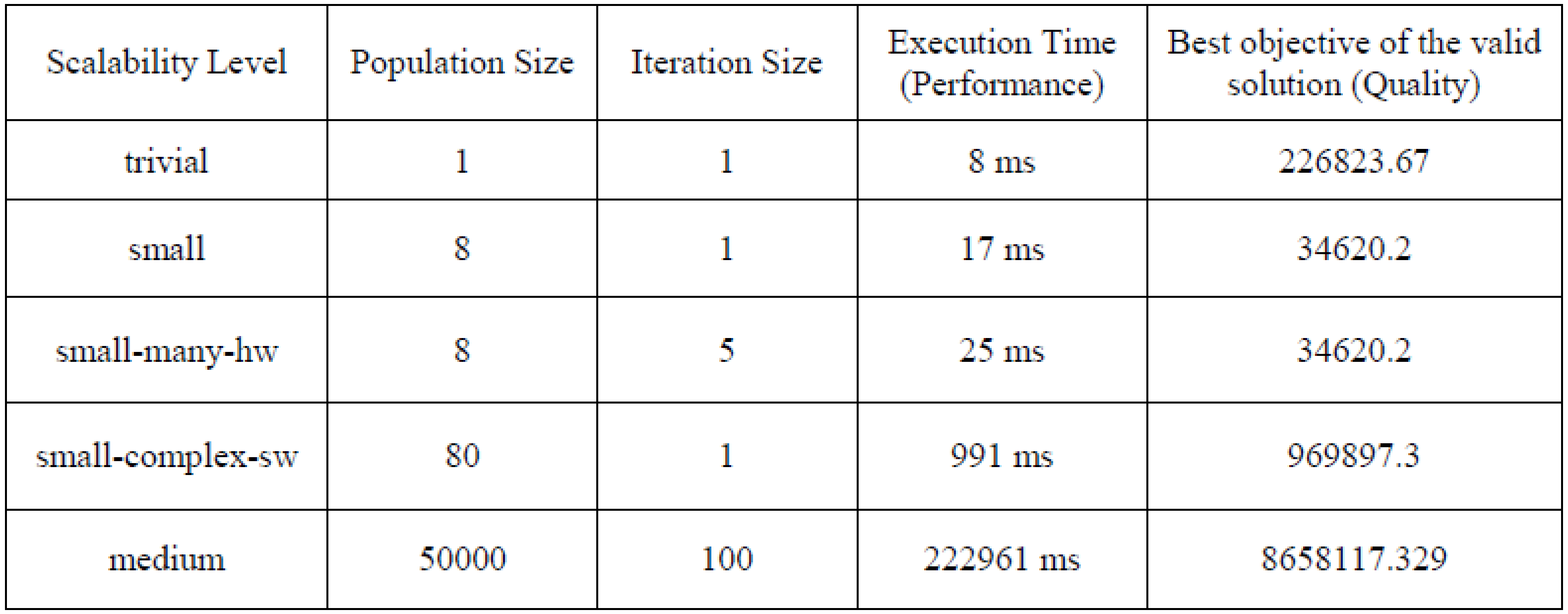
PART 2: FINDING THE BEST VALID SOLUTION

```
Initialize iteration size;
bestSolution = null;
For(int iteration=0; iteration< iteration size; iteration++){
    If time exceeds from maxSolvingTime: break;
    For each saved solution create an ant, and pass to current solution, possible resources, $\tau, \eta$ and P to it.
    Run all ants in parallel
    Update $\tau$ and P to new values
    If (bestSolution is null or antSolution.objective < bestSolution.objective)
        bestSolution = antSolution;
}
Return bestSolution;
```

RUNNING ANTS

```
Initialize $\alpha, \beta, \rho$ and Q;
Sort the assignments of the solution based on the number of possible resources
For each assignments i {
    if there is not any possible resource that has not been used by other assignment:
        return null;
    else:
    select a possible resource j using Rolette Wheel mechanism that has not been used by other assignment.
    Assign resource to the assignment
    $\tau_{ij} = \tau_{ij} + \frac{Q}{assignment.objective}$ ;
    $\tau_{ij} = (1-\rho) * \tau_{ij}$ ; //evaporation $\tau$ for the selected possible resource
    $P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{\forall j}(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$ ;
}
Return the solution;
```

RESULTS



Scalability Level	Population Size	Iteration Size	Execution Time (Performance)	Best objective of the valid solution (Quality)
trivial	1	1	8 ms	226823.67
small	8	1	17 ms	34620.2
small-many-hw	8	5	25 ms	34620.2
small-complex-sw	80	1	991 ms	969897.3
medium	50000	100	222961 ms	8658117.329

For larger benchmark ACO Solver meets the time out.

CONCLUSION

- The ACO algorithm is used to solve the Quality-based Software-Selection and Hardware-Mapping problem.
- The scalability, performance and quality of the proposed solution are tested on different benchmarks.
- The outcome indicates that this approach generates correct results for the evaluated test cases.
- Additionally, better results in terms of performance are given in comparison with the results of the ILP algorithm in the case description.