Version 1-3 Subsystem test

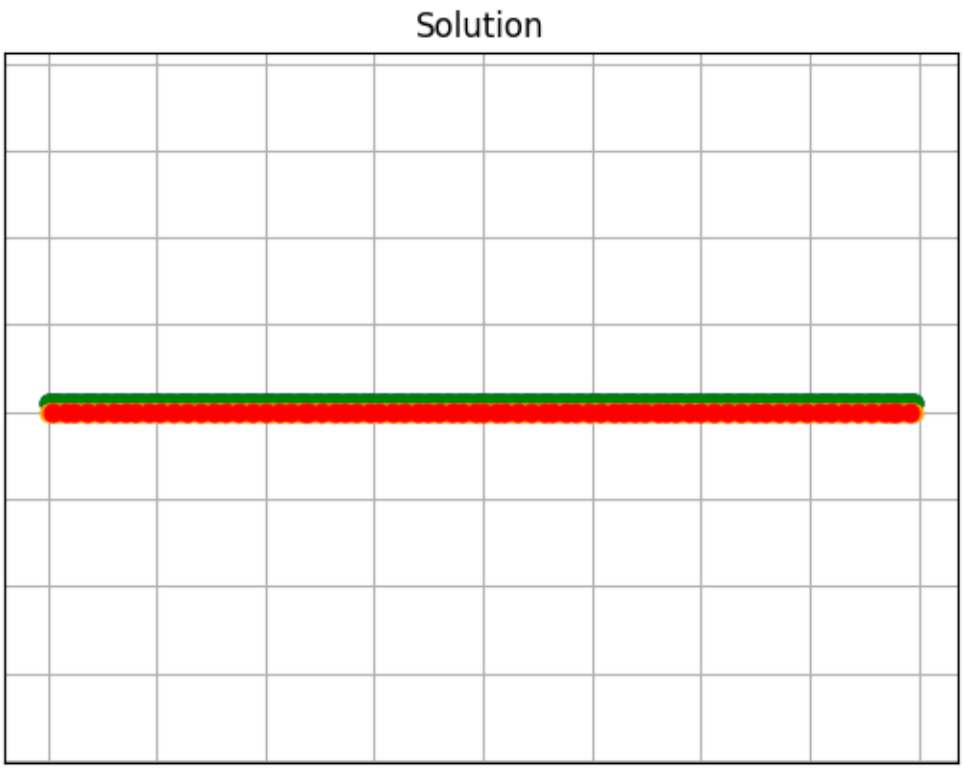
December 1, 2024

[3]:
```
import neal

from data.sp_data import SPData
from models import SPQuboBinary
from evaluation.evaluation import SPEvaluation
from plotting.sp_plot import SPPlot
```

[9]:
```
params = {"version": 1, "num_cols": 200, "rad_max": 2.4}
data = SPData().gen_problem(**params)
```

[10]:
```
while True:

    config = {"num_reads": 1500, "num_sweeps": 1500}
    solve_func = neal.SimulatedAnnealingSampler().sample_qubo
    qubo_model_bin = SPQuboBinary(data, P1=0.8, P3=1)
    answer = qubo_model_bin.solve(solve_func, **config)

    evaluation = SPEvaluation(data, answer["solution"])
    #print(f"solution clean: {evaluation.solution}")

    print(f"objective = {evaluation.get_objective()}")


    for constraint, violations in evaluation.check_solution().items():
        if len(violations) > 0:
            print(f"constraint {constraint} was violated {len(violations)}␣
 ↪times")
    print(len(violations))
    if len(violations)==0:
        break

plt = SPPlot(data, evaluation).plot_solution(hide_never_covered=True)
plt.show()
```

```
objective = 73
0
```



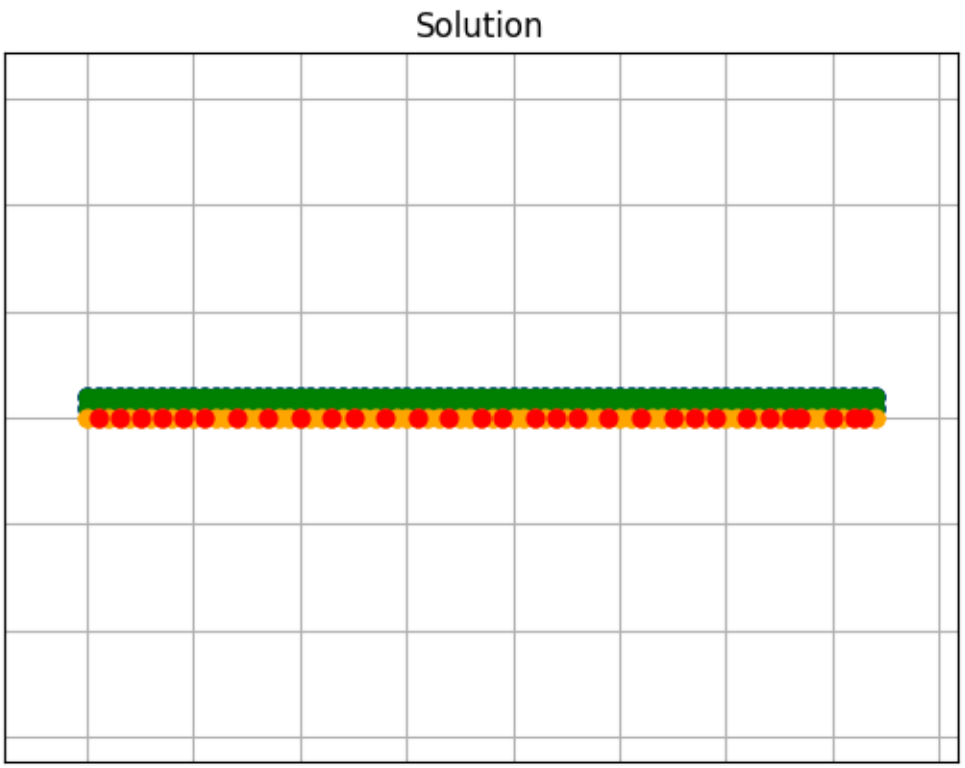
0.0.1 Version 2:

[6]:

```
while True:
    params = {"version": 2, "num_cols": 75, "rad_max": 2.4}
    data = SPData().gen_problem(**params)
    plt = SPPlot(data).plot_problem()
    # plt.show()

    config = {"num_reads": 1500, "num_sweeps": 1500}
    solve_func = neal.SimulatedAnnealingSampler().sample_qubo
    qubo_model_bin = SPQuboBinary(data, P1=0.8, P2=2, P3=1)
    answer = qubo_model_bin.solve(solve_func, **config)

    evaluation = SPEvaluation(data, answer["solution"])
    #print(f"solution clean: {evaluation.solution}")

    print(f"objective = {evaluation.get_objective()}")
    for constraint, violations in evaluation.check_solution().items():
```

```
        if len(violations) > 0:
            print(f"constraint {constraint} was violated {len(violations)}␣
 ↪times")
    print(len(violations))
    if len(violations)==0:
        break

plt = SPPlot(data, evaluation).plot_solution(hide_never_covered=True)
plt.show()
```

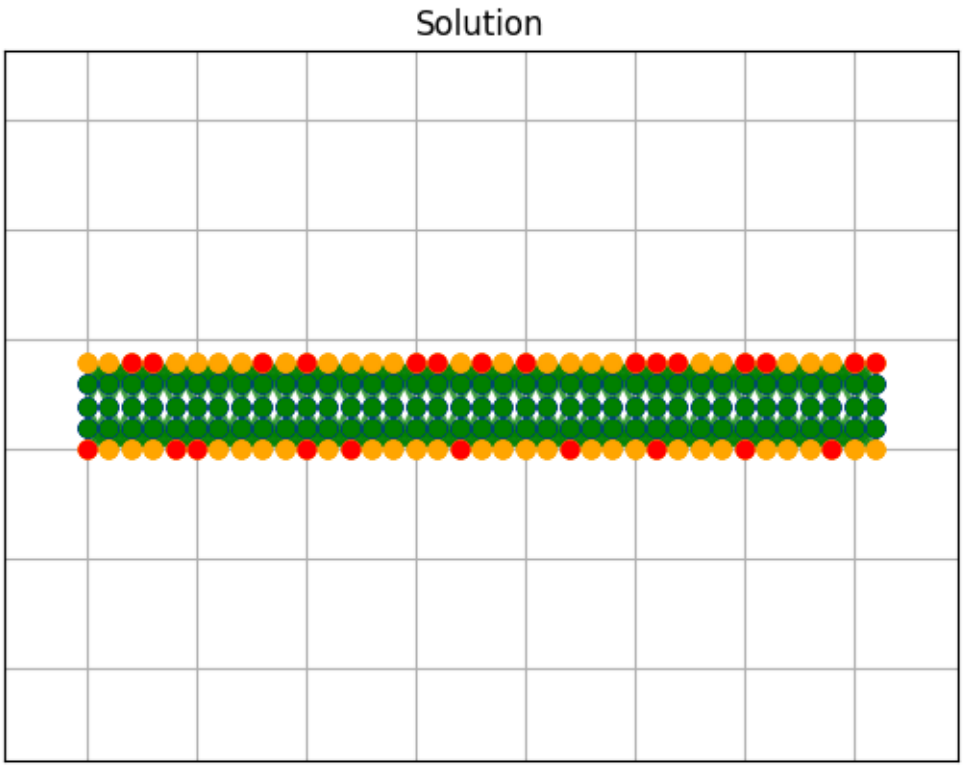
```
objective = 31
0
```



[7]:
```
while True:
    params = {"version": 3, "num_cols": 37, "rad_max": 2.4}
    data = SPData().gen_problem(**params)
    plt = SPPlot(data).plot_problem()
    # plt.show()

    config = {"num_reads": 1500, "num_sweeps": 1500}
```

```
    solve_func = neal.SimulatedAnnealingSampler().sample_qubo
    qubo_model_bin = SPQuboBinary(data, P1=0.8, P2=2, P3=1)
    answer = qubo_model_bin.solve(solve_func, **config)

    evaluation = SPEvaluation(data, answer["solution"])
    #print(f"solution clean: {evaluation.solution}")

    print(f"objective = {evaluation.get_objective()}")
    for constraint, violations in evaluation.check_solution().items():
        if len(violations) > 0:
            print(f"constraint {constraint} was violated {len(violations)}␣
 ↪times")
    print(len(violations))
    if len(violations)==0:
        break

plt = SPPlot(data, evaluation).plot_solution(hide_never_covered=True)
plt.show()
```

```
objective = 24
constraint missing_achievable_coverage was violated 1 times
1
objective = 25
0
```



Activated Lidars: 25
missing achievable coverage: []