Um algoritmo eficiente para um problema multiobjetivo de roteamento em rede de VANTs

Elias L. Marques Jr. Vitor N. Coelho Igor M. Coelho
Bruno N. Coelho Luiz Satoru Ochi

August 04, 2022

Summary

Summary

Introduction

Smart Cities

Cities that incorporate information and communication technologies (ICT) to improve the quality and performance of urban services.

Urban Services

- Communication
- Governance
- Security
- Energy
- Sustainability
- **Transport**

Drones

Reality

- Miniaturization of electronic control systems
- Electronic component cost reduction

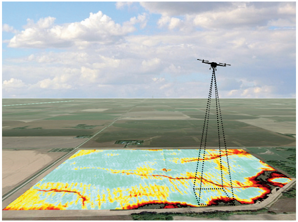
More than hobby, entertainment and photography!

Applications

Inspections [1], [2], [3] e [4]

Infrastructure and energy:

- Reduces risk of accidents
- Cost reduction
- Less invasive operations



Area monitoring

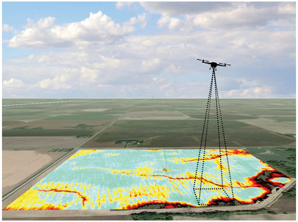
- Remote sensing data collection [5]
- Real-time mapping
- Autonomous navigation
- Environmental monitoring [6]

Applications

Inspections [1], [2], [3] e [4]

Infrastructure and energy:

- Reduces risk of accidents
- Cost reduction
- Less invasive operations



Area monitoring

- Remote sensing data collection [5]
- Real-time mapping
- Autonomous navigation
- Environmental monitoring [6]

Transport

Reality

- Google
- Amazon
- DHL
- UPS
- FedEx
- ...

Current solutions

1. **TSP**
2. **Routing**
   VRP
3. **Green Routing**
   G-VRP
4. **UAVs**
   TSPD
   VRPD
   UVRP

What are we looking for?

Fast + Eco + Dynamic = Perfect setting!

Goal

Establish routes that drones run on quickly, economically and continuously

Summary

MOGRDGP

Multi-Objective Green Routing Drone Grid Problem

MOGRDGP

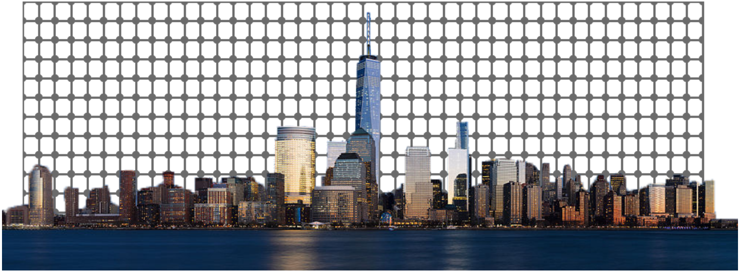
Establish UAV routes in an airspace, represented by a grid, visiting customers and avoiding no-go areas.

MOGRDGP

Multi-Objective Green Routing Drone Grid Problem

MOGRDGP

Establish UAV routes in an airspace, represented by a grid, visiting customers and avoiding no-go areas.



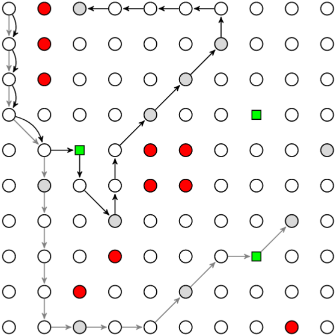
Definition

Goals

- Final Charge
- Time
- Consumption

Constraints

- Consumption
- Prohibited area



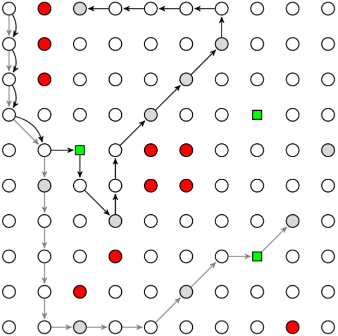
Definition

Goals

- Final Charge
- Time
- Consumption

Constraints

- Consumption
- Prohibited area



Summary

1. Introduction
2. MOGRDGP
3. Metaheuristics
   - Algorithm A*
   - G-MOVND
   - BRKGA
4. Experiments
5. Conclusions

Methods

VNS

1. Build
   - GRASP
2. Local Search
   - VND and MOVND

Genetic Algorithm

- BRKGA

Subpath

- A*

Methods

VNS

1. Build
   - GRASP
2. Local Search
   - VND and MOVND

Genetic Algorithm

- BRKGA

Subpath

- A*

Metaheuristics

- G-VND
- G-MOVND
- BRKGA

Summary

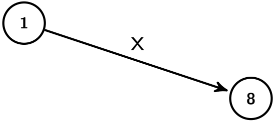
Algorithm

Sub-path

- The distance between two points in a **graphs** problem is predetermined
- Grid routing we need to calculate each **subroute**

A*

- Path tree
- The best path is determined by the lowest cost **f(n) = g(n) + h(n)**
- **g(n)** is the cost of the path from the start node to *n* and **h(n)** is a heuristic function that estimates the cost of the best path from *n* to the goal
- Chebyshev Distance



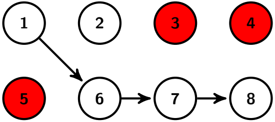
Algorithm

Sub-path

- The distance between two points in a **graphs** problem is predetermined
- Grid routing we need to calculate each **subroute**

A*

- Path tree
- The best path is determined by the lowest cost $\mathbf{f(n) = g(n) + h(n)}$
- $\mathbf{g(n)}$ is the cost of the path from the start node to n and $\mathbf{h(n)}$ is a heuristic function that estimates the cost of the best path from n to the goal
- Chebyshev Distance



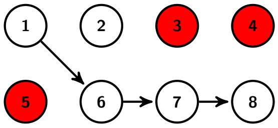
Algorithm

Sub-path

- The distance between two points in a **graphs** problem is predetermined
- Grid routing we need to calculate each **subroute**

A*

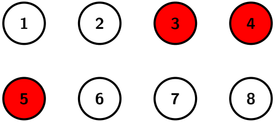
- Path tree
- The best path is determined by the lowest cost $\mathbf{f(n) = g(n) + h(n)}$
- $\mathbf{g(n)}$ is the cost of the path from the start node to n and $\mathbf{h(n)}$ is a heuristic function that estimates the cost of the best path from n to the goal
- Chebyshev Distance



Algorithm

A*

- Insert the initial node into *openSet*
- Until you reach your goal:
  - The first node of *openSet* is the node **current**
  - Removes node **current** from *openSet*
  - For every **neighbor**:
    - **tempG** = **g(current)** + **d(current, neighbor)**
    - If neighbor has **tempG** < **g(neighbor)**, **g(neighbor)** = **tempG**, update the other values and insert **neighbor** into *openSet*



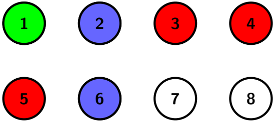
- current =
- openSet = []

	origin	f	g	h
1	-	3,16	0	3,16
2		∞	∞	2,24
6		∞	∞	2
7		∞	∞	1
8		∞	∞	0

Algorithm

A*

- Insert the initial node into *openSet*
- Until you reach your goal:
  - The first node of *openSet* is the node **current**
  - Removes node **current** from *openSet*
  - For every **neighbor**:
    - **tempG = g(current) + d(current, neighbor)**
    - If neighbor has **tempG** < **g(neighbor)**, **g(neighbor)** = **tempG**, update the other values and insert **neighbor** into *openSet*



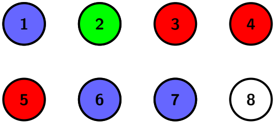
- current = 1
- openSet = [2, 6]

	origin	f	g	h
1	-	3,16	0	3,16
2	1	3,24	1	2,24
6	1	3	1	2
7		∞	∞	1
8		∞	∞	0

Algorithm

A*

- Insert the initial node into *openSet*
- Until you reach your goal:
  - The first node of *openSet* is the node **current**
  - Removes node **current** from *openSet*
  - For every **neighbor**:
    - **tempG = g(current) + d(current, neighbor)**
    - If neighbor has **tempG** < **g(neighbor)**, **g(neighbor)** = **tempG**, update the other values and insert **neighbor** into *openSet*



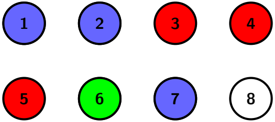
- current = 2
- openSet = [6, 7]

	origin	f	g	h
1	-	3,16	0	3,16
2	1	3,24	1	2,24
6	1	3	1	2
7	2	3	2	1
8		∞	∞	0

Algorithm

A*

- Insert the initial node into *openSet*
- Until you reach your goal:
  - The first node of *openSet* is the node **current**
  - Removes node **current** from *openSet*
  - For every **neighbor**:
    - **tempG** = **g(current)** + **d(current, neighbor)**
    - If neighbor has **tempG** < **g(neighbor)**, **g(neighbor)** = **tempG**, update the other values and insert **neighbor** into *openSet*



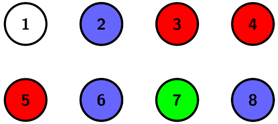
- current = 6
- openSet = [7]

	origin	f	g	h
1	-	3,16	0	3,16
2	1	3,24	1	2,24
6	1	3	1	2
7	2	3	2	1
8		∞	∞	0

Algorithm

A*

- Insert the initial node into *openSet*
- Until you reach your goal:
  - The first node of *openSet* is the node **current**
  - Removes node **current** from *openSet*
  - For every **neighbor**:
    - **tempG** = **g(current)** + **d(current, neighbor)**
    - If neighbor has **tempG** < **g(neighbor)**, **g(neighbor)** = **tempG**, update the other values and insert **neighbor** into *openSet*



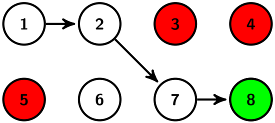
- current = 7
- openSet = [8]

	origin	f	g	h
1	-	3,16	0	3,16
2	1	3,24	1	2,24
6	1	3	1	2
7	2	3	2	1
8	7	3	3	0

Algorithm

A*

- Insert the initial node into *openSet*
- Until you reach your goal:
  - The first node of *openSet* is the node **current**
  - Removes node **current** from *openSet*
  - For every **neighbor**:
    - **tempG = g(current) + d(current, neighbor)**
    - If neighbor has **tempG** < **g(neighbor)**, **g(neighbor)** = **tempG**, update the other values and insert **neighbor** into *openSet*



- current = 8
- openSet = []

	origin	f	g	h
1	-	3,16	0	3,16
2	1	3,24	1	2,24
6	1	3	1	2
7	2	3	2	1
8	7	3	3	0

Summary

Basic Algorithm

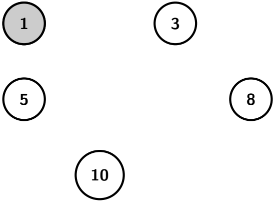
Algorithm 1 G-VND

1: **repeat**
2: $E \leftarrow \{\}$
3: $s_i \leftarrow GRASPBuilder()$
4: $E \leftarrow Update(E, s_i)$
5: $E \leftarrow MOVND(E, Neighborhood)$
6: **until** time does not end
7: **return** E

GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

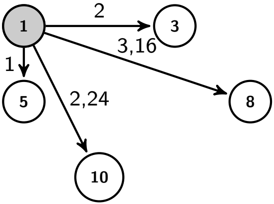
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

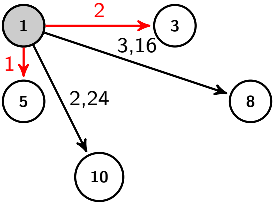
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to
    its distance from r
    Updates RCL considering only α% best
    CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

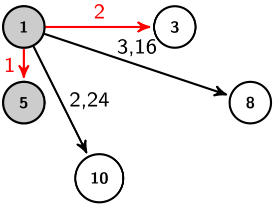
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to
    its distance from r
    Updates RCL considering only α% best
    CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

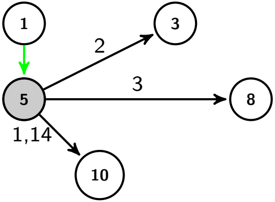
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

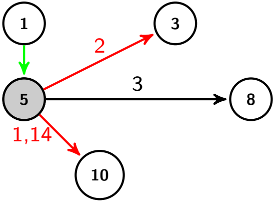
- $\alpha = 2$



GRASP

$o \leftarrow$ random origin
$s \leftarrow s \cup \{o\}$
Initialize Candidate List CL
if o is a client **then**
    $CL \leftarrow CL - \{o\}$
end if
$r \leftarrow o$
while $CL \neq$ **do**
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c $\in$ RCL randomly
    $s \leftarrow s \cup \{c\}$
    $r \leftarrow c$
    $CL \leftarrow CL - \{r\}$
end while
return s

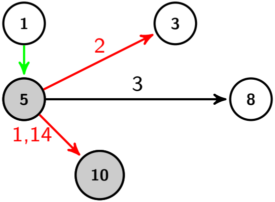
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to
    its distance from r
    Updates RCL considering only α% best
    CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

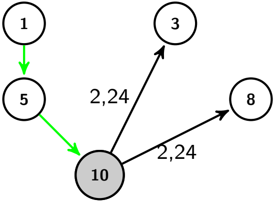
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

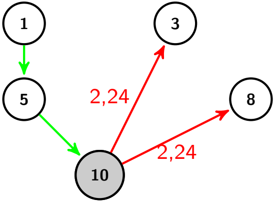
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

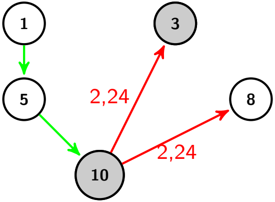
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

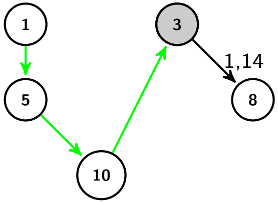
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

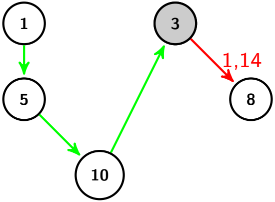
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

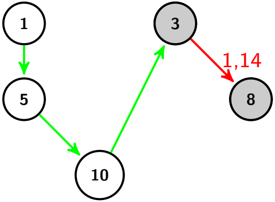
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to its distance from r
    Updates RCL considering only α% best CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

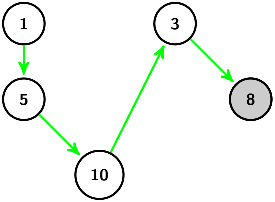
- $\alpha = 2$



GRASP

```
o ← random origin
s ← s ∪ {o}
Initialize Candidate List CL
if o is a client then
    CL ← CL − {o}
end if
r ← o
while CL ≠ do
    Sort CL in ascending order according to
    its distance from r
    Updates RCL considering only α% best
    CL candidates
    Choose c ∈ RCL randomly
    s ← s ∪ {c}
    r ← c
    CL ← CL − {r}
end while
return s
```

- $\alpha = 2$



Local Search

MOVND

- While **localPool** is not empty:
  - $S \leftarrow$ **localPool**$[0]$
  - Remove S from **localPool** and insert in **globalPool**
  - For all neighborhood N:
    - $S' \leftarrow N(S)$
    - If S' dominates S: insert S' in **localPool** and reset N to the first neighborhood

Local Search

MOVND

- While **localPool** is not empty:
  - $S \leftarrow$ **localPool**$[0]$
  - Remove S from **localPool** and insert in **globalPool**
  - For all neighborhood N:
    - $S' \leftarrow N(S)$
    - If S' dominates S: insert S' in **localPool** and reset N to the first neighborhood

VND

- For all neighborhood N:
  - $S' \leftarrow N(S)$
  - If $f(S') < f(S)$:
    - $S \leftarrow S'$
    - Reset N to the first neighborhood

Neighborhoods

Intraroute

- Swap(1,1) - S1, S2, S3, S4
- Remove Recharge Point
- Nearest Recharge Point
- Remove Repeated
- Section Speed Increase
- Section Speed Decrease
- Random Increase in Recharge Rate
- Random Decrease in Recharge Rate
- Random Speed Increase
- Random Speed Decrease

Inter-route

- Swap(1,1)
- Shift(1,0)

Acceptance Criteria

Multiobjective

- Pareto front
- Dominance

Mono-objective

- Pool size equals 1
- Fitness Function

$$f(x) = t(x) + c(x) - 5 * cf(x)$$

Summary

BRKGA

Biased Random Key Genetic Algorithm

Genetic Algorithm

- Crossover
- Mutation

Random Key

- A random key is a random real number in the continuous range $[0, 1)$
- A **decoder** is a deterministic algorithm that takes a vector of random keys as input and returns a solution to the optimization problem

BRKGA

Biased Random Key Genetic Algorithm

Genetic Algorithm

- Crossover
- Mutation

Random Key

- A random key is a random real number in the continuous range $[0, 1)$
- A **decoder** is a deterministic algorithm that takes a vector of random keys as input and returns a solution to the optimization problem

Structure

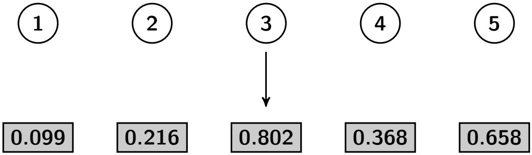
BRKGA

- The initial population consists of n vectors with c random keys each
- The first e individuals (**elite**) are kept in the population, as well as other m random individuals (**mutation**)
- The **crossover** is a cross between a solution from the elite population (with factor ρ - **biased**) with another solution from the population to generate a child
- Bean [7] proposed **decoders** based on sorting the vector of random keys to produce a sequence

Structure

BRKGA

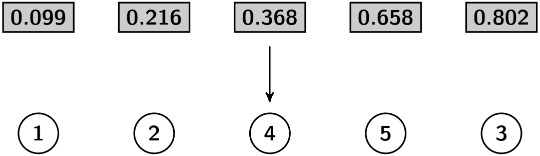
- The initial population consists of n vectors with c random keys each
- The first e individuals (**elite**) are kept in the population, as well as other m random individuals (**mutation**)
- The **crossover** is a cross between a solution from the elite population (with factor ρ - **biased**) with another solution from the population to generate a child
- Bean [7] proposed **decoders** based on sorting the vector of random keys to produce a sequence



Structure

BRKGA

- The initial population consists of n vectors with c random keys each
- The first e individuals (**elite**) are kept in the population, as well as other m random individuals (**mutation**)
- The **crossover** is a cross between a solution from the elite population (with factor ρ - **biased**) with another solution from the population to generate a child
- Bean [7] proposed **decoders** based on sorting the vector of random keys to produce a sequence



Implementation

Stages

- First stage: we look at the problem as a graph routing problem
- Second stage: decoding is now integrated into method A*

Individual

- Each individual is represented by three random key vectors: visitation order, speed and vehicle recharge rate

Rank

- Fitness function used in the G-VND method

Implementation

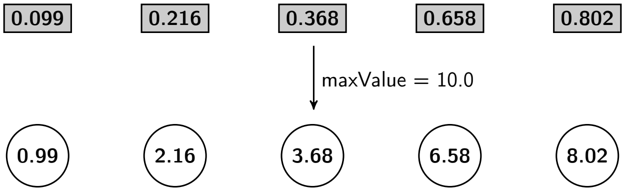
Decoding

- Visitation Order: follows the basic principle of RKGA applied to routing. In this way, encoding and decoding is performed by sorting the keys
- Speed and Recharge Rate: decoding works by multiplying the value of the random key by the maximum value of the variable. So, at the end of the decoding, we have an array of speeds and recharge rates for each leg of the route

Implementation

Decoding

- Visitation Order: follows the basic principle of RKGA applied to routing. In this way, encoding and decoding is performed by sorting the keys
- Speed and Recharge Rate: decoding works by multiplying the value of the random key by the maximum value of the variable. So, at the end of the decoding, we have an array of speeds and recharge rates for each leg of the route



Summary

Configurations

Environment

- Algorithms implemented in C++
- Virtual machine with 2 GB of virtual RAM with Windows 10 as the host OS.
- Intel Core i5-6400 CPU with 16 GB of RAM
- Ubuntu 18.04 64-bit

Instances

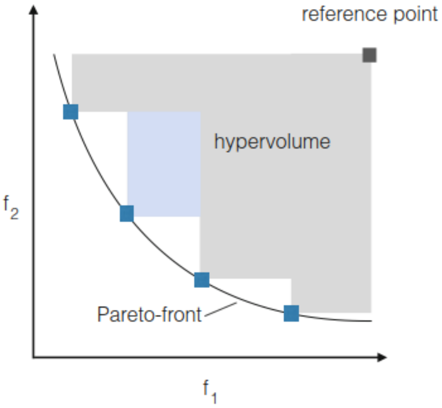
Features

- eil51, eil101 and rat195
- Origin point
- Number of UAVs
- Variable consumption (c_v)
- Limit time
  - 51 clients: 5s, 10s, 30s, 60s, 120s, e 300s (default)
  - 101 clients: 10s, 30s, 60s, 120s, 300s e 600s (default)
  - 195 clients: 900s (default) e 1800s
- Preprocessing
- 92 Instances

Comparison

Measurements

- Hipervolume
- Coverage



Comparison

Measurements

- Hipervolume
- Coverage

Algorithm 2 Coverage

```
solDominateds ← 0
for a ∈ Pareto do
    for b ∈ CurrentSet do
        if a.weaklyDominates(b) then
            solDominateds ← solDominateds + 1
            break
        end if
    end for
end for
return solDominateds / size(Pareto)
```

Results

Table 1: Comparison of objective function values in standard instances

	O1			O2			O3		
Instance	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND
eil51a1_pp_1d_005_300	**99**	**99**	**99**	989	**542**	570	466	268	**121**
eil51a2_pp_1d_005_300	98	97	**99**	1094	**588**	601	523	268	**78**
eil51b1_pp_1d_005_300	97	98	**99**	1102	**543**	560	539	265	**179**
eil51b2_pp_1d_005_300	92	98	**99**	1119	**547**	570	548	263	**183**
eil101a1_pp_1d_005_600	86	98	**99**	1755	**943**	1078	598	453	**275**
eil101a2_pp_1d_005_600	89	**99**	**99**	1646	**915**	1130	580	434	**264**
eil101b1_pp_1d_005_600	94	**99**	**99**	1381	**942**	989	665	442	**181**
eil101b2_pp_1d_005_600	95	97	**99**	1907	**922**	989	604	432	**181**
rat195a1_pp_1d_005_900	**94**	-	-	**59055**	-	-	**9827**	-	-
rat195a2_pp_1d_005_900	**99**	-	-	**87504**	-	-	**10866**	-	-
rat195b1_pp_1d_005_900	**89**	-	-	**97415**	-	-	**11559**	-	-
rat195b2_pp_1d_005_900	**39**	-	-	**63650**	-	-	**9449**	-	-
rat195a1_pp_1d_005_1800	**94**	-	-	**91888**	-	-	**7677**	-	-
rat195a2_pp_1d_005_1800	**96**	-	-	**123963**	-	-	**12201**	-	-
rat195b1_pp_1d_005_1800	**87**	-	-	**95716**	-	-	**9707**	-	-
rat195b2_pp_1d_005_1800	**46**	-	-	**144515**	-	-	**12572**	-	-
Victories/Draws	**9**	3	8	**8**	**8**	0	**8**	0	**8**

Results

Table 2: Comparison of objective function values in instances without preprocessing

Instance	O1 BRKGA	O1 G-VND	O1 G-MOVND	O2 BRKGA	O2 G-VND	O2 G-MOVND	O3 BRKGA	O3 G-VND	O3 G-MOVND
eil51a1_1d_005_300	90	**99**	**99**	1056	**562**	584	511	269	**145**
eil51a2_1d_005_300	91	98	**99**	1238	599	**588**	596	287	**168**
eil51b1_1d_005_300	99	98	**99**	1269	574	**535**	616	275	**114**
eil51b2_1d_005_300	92	**99**	**99**	1466	596	**595**	717	286	**237**
eil101a1_1d_005_600	81	98	**99**	1487	**981**	1059	749	477	**198**
eil101a2_1d_005_600	98	**99**	**99**	3018	**981**	1276	1461	463	**221**
eil101b1_1d_005_600	94	**99**	**99**	1442	1107	**1039**	710	528	**404**
eil101b2_1d_005_600	66	95	**99**	1924	**1092**	1116	945	509	**312**
rat195a1_1d_005_900	**69**	-	-	**155419**	-	-	**10323**	-	-
rat195a2_1d_005_900	-	-	-	-	-	-	-	-	-
rat195b1_1d_005_900	**96**			**184292**			**10628**		
rat195b2_1d_005_900	-	-	-	-	-	-	-	-	-
Victories/Empate	2	4	**8**	2	**4**	**4**	2	0	**8**

Results

Table 3: Comparison of objective function values in instances with 2 drones

Instance	O1 BRKGA	O1 G-VND	O1 G-MOVND	O2 BRKGA	O2 G-VND	O2 G-MOVND	O3 BRKGA	O3 G-VND	O3 G-MOVND
eil51a1_pp_2d_005_300	85	95	**98**	804	**267**	297	659	239	**152**
eil51a2_pp_2d_005_300	91	92	**98**	1066	**258**	261	918	**238**	241
eil51b1_pp_2d_005_300	98	98	**99**	867	**266**	**266**	778	242	**203**
eil51b2_pp_2d_005_300	89	97	**99**	1115	279	**278**	968	238	**152**
eil101a1_pp_2d_005_600	83	**51**	55	2359	**723**	728	2221	**620**	677
eil101a2_pp_2d_005_600	86	**91**	**91**	3342	**665**	781	3090	**595**	697
eil101b1_pp_2d_005_600	87	94	**98**	2220	**523**	528	2073	472	**413**
eil101b2_pp_2d_005_600	75	90	**94**	3241	**538**	703	2565	**506**	639
rat195a1_pp_2d_005_900	**44**	-	-	**73438**	-	-	**11938**	-	-
rat195a2_pp_2d_005_900	-	-	-	-	-	-	-	-	-
rat195b1_pp_2d_005_900	**63**	-	-	**134721**	-	-	**13887**	-	-
rat195b2_pp_2d_005_900	**34**			**108718**			**14632**		
Victories/Empate	3	2	**7**	3	**7**	2	3	**4**	**4**

Results

Table 4: Comparison of objective function values in eil51 instances with 2 drones and c_v equals to 0.1

Instance	O1			O2			O3		
	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND	BRKGA	G-VND	G-MOVND
eil51a1_pp_2d_010_300	82	98	**99**	1229	288	**275**	435	407	**243**
eil51a2_pp_2d_010_300	84	**98**	97	2417	**283**	285	488	416	**159**
eil51b1_pp_2d_010_300	80	**99**	**99**	1454	**292**	309	542	445	**207**
eil51b2_pp_2d_010_300	60	97	**98**	1145	311	**292**	714	420	**360**
eil51a1_pp_2d_010_120	88	97	**99**	2013	310	**292**	494	515	**225**
eil51a2_pp_2d_010_120	67	**98**	91	17411	319	**293**	442	540	**239**
eil51b1_pp_2d_010_120	65	**99**	**99**	1744	301	**295**	535	499	**232**
eil51b2_pp_2d_010_120	65	98	**99**	1159	418	**328**	822	575	**95**
eil51a1_pp_2d_010_60	74	**99**	95	2466	339	**338**	526	366	**288**
eil51a2_pp_2d_010_60	73	**92**	**92**	3703	375	**313**	639	590	**201**
eil51b1_pp_2d_010_60	48	97	**99**	3071	344	**319**	549	556	**263**
eil51b2_pp_2d_010_60	66	95	**98**	1087	**349**	370	835	558	**214**
eil51a1_pp_2d_010_30	70	-	**96**	171691	-	**344**	574	-	**245**
eil51a2_pp_2d_010_30	**82**	-	76	3254	-	**312**	869	-	**525**
eil51b1_pp_2d_010_30	67	-	**97**	1925	-	**359**	841	-	**385**
eil51b2_pp_2d_010_30	50	-	**97**	1788	-	**347**	1111	-	**282**
eil51a1_pp_2d_010_10	**83**	-	-	**3694**	-	-	**960**	-	-
eil51a2_pp_2d_010_10	**61**	-	-	**46142**	-	-	**889**	-	-
eil51b1_pp_2d_010_10	**42**	-	-	**3104**	-	-	**1145**	-	-
eil51b2_pp_2d_010_10	**44**	-	-	**3249**	-	-	**1077**	-	-
eil51a1_pp_2d_010_5	**86**	-	-	**5612**	-	-	**1055**	-	-
eil51a2_pp_2d_010_5	**72**	-	-	**12997**	-	-	**1178**	-	-
eil51b1_pp_2d_010_5	**18**	-	-	**6813**	-	-	**1110**	-	-
eil51b2_pp_2d_010_5	**10**	-	-	**6642**	-	-	**1142**	-	-
Victories/Draws	9	6	**12**	8	3	0	8	0	**13**

Results

Table 5: Comparison of objective function values in eil101 instances with 2 drones and c_v equals to 0.1

Instance	O1 BRKGA	O1 G-VND	O1 G-MOVND	O2 BRKGA	O2 G-VND	O2 G-MOVND	O3 BRKGA	O3 G-VND	O3 G-MOVND
eil101a1_pp_2d_010_600	**75**	-	-	**4275**	-	-	**1279**	-	-
eil101a2_pp_2d_010_600	**77**	-	-	**3327**	-	-	**13882**	-	-
eil101b1_pp_2d_010_600	**94**	-	-	**2844**	-	-	**1298**	-	-
eil101b2_pp_2d_010_600	**88**	-	-	**3382**	-	-	**1485**	-	-
eil101a1_pp_2d_010_300	**70**	-	-	**3415**	-	-	**1538**	-	-
eil101a2_pp_2d_010_300	**70**	-	-	**5681**	-	-	**1591**	-	-
eil101b1_pp_2d_010_300	**80**	-	-	**2602**	-	-	**1826**	-	-
eil101b2_pp_2d_010_300	**83**	-	-	**5456**	-	-	**1599**	-	-
eil101a1_pp_2d_010_120	**94**	-	-	**5734**	-	-	**1751**	-	-
eil101a2_pp_2d_010_120	**59**	-	-	**9791**	-	-	**1928**	-	-
eil101b1_pp_2d_010_120	**63**	-	-	**5469**	-	-	**2894**	-	-
eil101b2_pp_2d_010_120	**76**	-	-	**3663**	-	-	**2480**	-	-
eil101a1_pp_2d_010_60	**95**	-	-	**6853**	-	-	**1973**	-	-
eil101a2_pp_2d_010_60	**40**	-	-	**16252**	-	-	**1859**	-	-
eil101b1_pp_2d_010_60	**97**	-	-	**11621**	-	-	**2070**	-	-
eil101b2_pp_2d_010_60	**58**	-	-	**6563**	-	-	**2245**	-	-
eil101a1_pp_2d_010_30	**29**	-	-	**12176**	-	-	**2336**	-	-
eil101a2_pp_2d_010_30	**7**	-	-	**19280**	-	-	**2166**	-	-
eil101b1_pp_2d_010_30	**63**	-	-	**9231**	-	-	**2543**	-	-
eil101b2_pp_2d_010_30	**46**	-	-	**7188**	-	-	**2699**	-	-
eil101a1_pp_2d_010_10	**46**	-	-	**23792**	-	-	**2945**	-	-
eil101a2_pp_2d_010_10	**2**	-	-	**76780**	-	-	**2855**	-	-
eil101b1_pp_2d_010_10	**70**	-	-	**231696**	-	-	**2961**	-	-
eil101b2_pp_2d_010_10	**56**	-	-	**11341**	-	-	**3398**	-	-
Victories/Draws	**24**	0	0	**24**	0	0	**24**	0	0

Results

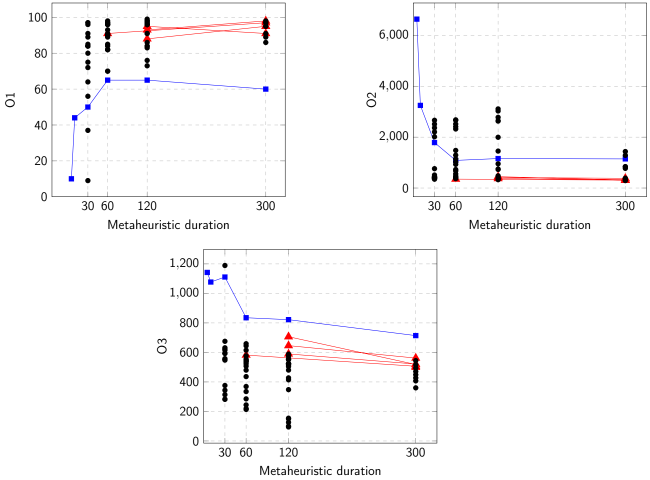


Figure: eil51b2 - BRKGA (blue), GMOVND (black) and GVND (red)

Results

Table 6: Comparison of hypervolume values in standard instances

	G-VND					G-MOVND				
Instance	S1	S2	S2-S3	S2-S4	S4	S1	S2	S2-S3	S2-S4	S4
eil51a1_pp_1d_005_300	0.020795	2e-06	0.030861	**0.023091**	0.016475	0.134317	0.061298	0.091998	0.15867	**0.211702**
eil51a2_pp_1d_005_300	0.010786	0.000325	0.005431	**0.057932**	0.013954	0.149194	0.020133	0.101524	0.077179	**0.18835**
eil51b1_pp_1d_005_300	0.025811	0.004155	0.019824	0.010168	**0.06474**	0.108522	0.049915	0.107278	0.167067	**0.220386**
eil51b2_pp_1d_005_300	0.025261	0.001832	0.007746	**0.078744**	0.058592	0.239336	0.133163	0.115436	0.169022	**0.29454**
eil101a1_pp_1d_005_600	0	0.154562	0.128263	**0.174743**	0.159414	0.079605	**0.441809**	0.172888	0.255015	0.2897
eil101a2_pp_1d_005_600	1e-06	0.057373	0.009536	0.081314	**0.105031**	0.03035	0.20937	0.136948	0.332207	**0.372734**
eil101b1_pp_1d_005_600	0.004005	0.059317	0.025549	0.175163	**0.196375**	0.002264	0.190141	0.209588	**0.388912**	0.380655
eil101b2_pp_1d_005_600	8e-06	0.132503	0.20232	0.190885	**0.239937**	0.002315	0.350836	0.298897	0.403424	**0.40505**
Victories/Draws	0	0	0	**4**	**4**	0	1	0	1	**6**

Results

Table 7: Comparison of hypervolume values in instances without preprocessing

Instance	G-VND S1	G-VND S2	G-VND S2-S3	G-VND S2-S4	G-VND S4	G-MOVND S1	G-MOVND S2	G-MOVND S2-S3	G-MOVND S2-S4	G-MOVND S4
eil51a1_1d_005_300	0.00257	0.000583	0.001212	0.016974	**0.044605**	0.079965	0.145023	0.092194	0.18989	**0.20687**
eil51a2_1d_005_300	0.000739	0.001257	0.003751	0.05076	**0.063572**	0.00175	0.127341	0.245604	0.139505	**0.288231**
eil51b1_1d_005_300	0.00582	1e-06	0.006716	0.012041	**0.013661**	0.076428	0.231991	0.002398	**0.156403**	0.141462
eil51b2_1d_005_300	0.001511	0.000914	0.016922	0.045074	**0.05892**	0.174241	0.038657	0.171226	0.21436	**0.26774**
eil101a1_1d_005_600	0	0.013875	0.184596	0.204001	**0.257515**	0.097376	**0.445235**	0.372797	0.321633	0.244023
eil101a2_1d_005_600	0	0.067694	0.046191	**0.157666**	0.13613	0.050911	0.002937	0.069462	**0.398372**	0.323383
eil101b1_1d_005_600	0.002451	0.134832	0.087889	0.050036	**0.14625**	0	0.0157	0.110997	**0.163167**	0.138303
eil101b2_1d_005_600	0	0.116751	0.156813	**0.207567**	0.014252	0	0.100817	0.20806	0.33971	**0.375293**
Victories/Draws	0	0	0	2	**6**	0	1	0	3	**4**

Results

Table 8: Comparison of hypervolume values in instances with 2 drones

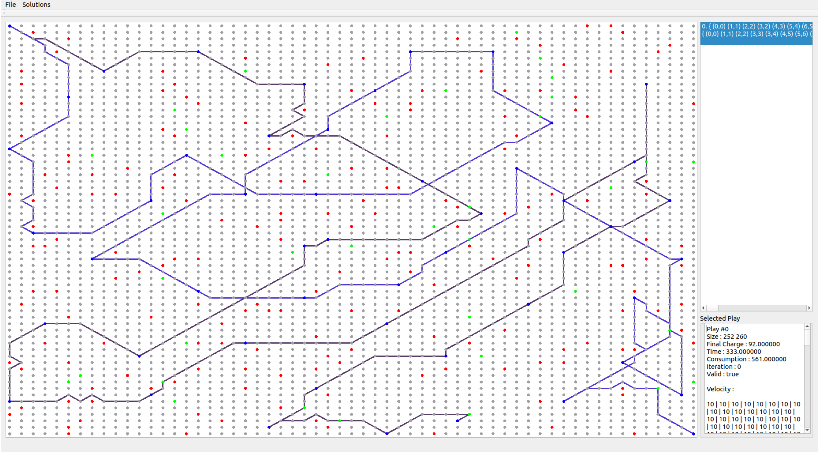
Instance	G-VND					G-MOVND				
	S1	S2	S2-S3	S2-S4	S4	S1	S2	S2-S3	S2-S4	S4
eil51a1_pp_2d_005_300	**<u>0.033723</u>**	0.021324	0.008385	0.016154	0.000514	0.319313	0.278176	0.127342	**<u>0.333434</u>**	0.282017
eil51a2_pp_2d_005_300	0.046482	**<u>0.067863</u>**	0.032031	0.069292	0.02343	0.155661	0.092021	0.121542	**<u>0.194389</u>**	0.057058
eil51b1_pp_2d_005_300	**<u>0.017558</u>**	0.007942	0.002682	0.00724	0.003639	0.220793	0.142356	**<u>0.239217</u>**	0.172656	0.067615
eil51b2_pp_2d_005_300	**<u>0.01701</u>**	0.000464	0.005443	0.000128	0.01946	**<u>0.302163</u>**	0.185784	0.267395	0.219764	0.211709
eil101a1_pp_2d_005_600	0.023204	0.013148	**<u>0.046031</u>**	0.007826	0.003264	0.007485	0.042657	0.010769	**<u>0.119539</u>**	0.040367
eil101a2_pp_2d_005_600	0.021231	0.012546	**<u>0.037505</u>**	0.041488	0.001901	**<u>0.082906</u>**	0.047043	0.052569	0.022703	0.054758
eil101b1_pp_2d_005_600	0.142191	0.057098	0.051016	**<u>0.152692</u>**	0.002819	0	0.097262	**<u>0.269367</u>**	0.258223	0.094262
eil101b2_pp_2d_005_600	**<u>0.032291</u>**	0.01123	1e-06	0.000647	1e-06	**<u>0.122753</u>**	0.073414	0.01136	0.065715	0.022717
Victories/Draws	**<u>4</u>**	1	2	1	0	**<u>3</u>**	0	2	**<u>3</u>**	0

Results

Table 9: Comparison of hypervolume values in eil51 instances with 2 drones and c_v equal to 0.1

Instance	G-VND					G-MOVND				
	S1	S2	S2-S3	S2-S4	S4	S1	S2	S2-S3	S2-S4	S4
eil51a1_pp_2d_010_300	**0.069485**	0.021716	0.015454	0.062854	0.000548	0.131346	**0.281926**	0.149923	0.199639	0.110301
eil51a2_pp_2d_010_300	**0.099269**	0.06019	0.079028	0.030441	0.000709	0.025053	0.124247	0.132469	**0.245771**	0.213089
eil51b1_pp_2d_010_300	0.065252	**0.081112**	0.039958	0.020731	0.044003	0.268067	0.290394	0.223338	0.284001	**0.299283**
eil51b2_pp_2d_010_300	**0.06844**	0.065438	0.06374	0.054796	0.026421	**0.253256**	0.095927	0.158712	0.154838	0.230625
eil51a1_pp_2d_010_120	**0.057691**	0.039874	0.028219	0.004429	0.011841	0.249306	0.097837	**0.387598**	0.304051	0.257775
eil51a2_pp_2d_010_120	0	0.003552	0.000195	**0.014564**	0.020997	**0.502308**	0.330587	0.161815	0.50082	0.400457
eil51b1_pp_2d_010_120	0.022257	0.007968	0.010326	**0.038498**	1e-06	**0.255684**	0.24811	0.18098	0.184642	0.241851
eil51b2_pp_2d_010_120	**0.069421**	0.015079	0.020671	0.004019	1e-06	**0.488216**	0.231857	0.137772	0.37673	0.244011
eil51a1_pp_2d_010_60	0.200294	**0.24779**	0.122142	0.150754	0.136076	0.141028	**0.213526**	0.162016	0.15151	0.167547
eil51a2_pp_2d_010_60	**0.234699**	0.170401	0.105089	0.003862	0	**0.44086**	0.431543	0.37044	0.128351	0.355009
eil51b1_pp_2d_010_60	0.005943	0.052966	0.051464	**0.059508**	0.001448	**0.313255**	0.111483	0.17727	0.038938	0.15267
eil51b2_pp_2d_010_60	0.083241	**0.122473**	2e-06	0.118052	0.000142	0.020816	0.054042	0.144838	0.159713	**0.311948**
eil51a1_pp_2d_010_30	-	-	-	-	-	0	**0.092966**	0.028953	0	0.073083
eil51a2_pp_2d_010_30	-	-	-	-	-	**0.049724**	0.027657	0.023998	0.018424	0.0079
eil51b1_pp_2d_010_30	-	-	-	-	-	0	0.03002	**0.061249**	0.036549	0.000435
eil51b2_pp_2d_010_30	-	-	-	-	-	0.37616	0.365801	0.369478	**0.398487**	0.159026
Victories/Draws	**6**	3	0	3	0	**7**	3	2	2	2

GUI


File Solutions
0. { (0,0) (1,1) (2,2) (3,2) (4,3) (5,4) (6,5
{ (0,0) (1,1) (2,2) (3,3) (3,4) (4,5) (5,6) (
Selected Play
Play #0
Size : 252 260
Final Charge : 92.000000
Time : 333.000000
Consumption : 561.000000
Iteration : 0
Valid : true
Velocity :
10 | 10 | 10 | 10 | 10 | 10 | 10 | 10
| 10 | 10 | 10 | 10 | 10 | 10 | 10 |
10 | 10 | 10 | 10 | 10 | 10 | 10 | 10
| 10 | 10 | 10 | 10 | 10 | 10 | 10 |

Summary

1. Introduction
2. MOGRDGP
3. Metaheuristics
   - Algorithm A*
   - G-MOVND
   - BRKGA
4. Experiments
5. Conclusions

Conclusions

- MOGRDGP: graphs X grids with docking constraints
- Metaheuristics
- MILP Algorithm
- Future Works:
    - Hybrid Algorithms
    - Real-world implementation

References

[1] Najib Metni and Tarek Hamel. "A UAV for bridge inspection: Visual servoing control law with orientation limits". In: *Automation in construction* 17.1 (2007), pp. 3–10.

[2] Koppány Máthé and Lucian Bușoniu. "Vision and control for UAVs: A survey of general methods and of inexpensive platforms for infrastructure inspection". In: *Sensors* 15.7 (2015), pp. 14887–14916.

[3] Geraldo José Adabo. "Long Range Unmanned Aircraft System for Power Line Inspection of Brazilian Electrical System". In: *Journal of Energy and Power Engineering* 8.2 (2014).

[4] Chung Deng et al. "Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications". In: *J. Commun* 9.9 (2014), pp. 687–692.

References

[5] Norbert Haala et al. "Performance test on UAV-based photogrammetric data collection". In: *Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38.1/C22 (2011), pp. 7–12.

[6] Alan Harris et al. "Alignment and tracking of a free-space optical communications link to a UAV". In: *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*. Vol. 1. IEEE. 2005, pp. 1–C.

[7] James C Bean. "Genetic algorithms and random keys for sequencing and optimization". In: *ORSA journal on computing* 6.2 (1994), pp. 154–160.

Um algoritmo eficiente para um problema multiobjetivo de roteamento em rede de VANTs

Elias L. Marques Jr. Vitor N. Coelho Igor M. Coelho
Bruno N. Coelho Luiz Satoru Ochi

August 04, 2022