HEURISTIC APPROACH FOR PATHS' COMPUTING OF A MARINE FLEET FOR AN UNDERWATER PATROLLING MISSION

D. Balestrieri [(a)], G. Vaccaro [(b)], F. Lancia[(b)] , F. Portieri[(a)]

[(a)] INTECS S.p.A. – Traffic Control Division - Salita del Poggio Laurentino 7 – 00144 Roma (Italy) - www.intecs.it
[(b)] Università degli Studi Roma Tre – Dipartimento di Informatica e Automazione – Via della Vasca Navale 79 – 00146 – Roma (Italy) - www.dia.uniroma3.it

[(a)]domenico.balestrieri@intecs.it, [(b)] giu.vaccaro@stud.uniroma3.it, [(c)] fra.lancia@stud.uniroma3.it, [(d)]fabio.portieri@intecs.it

ABSTRACT

Nowadays, the hypothesis of using low cost "unmanned" vehicles, to replace men into territorial waters' surveillance operations, is taking over. This would allow few operators to patrol great sea areas, reducing consequently costs of patrolling missions led by human resources. Because of low autonomy of robotic vehicles, compared with the autonomy of normal vehicles, path's planning algorithms are needed to maximize the length of each path, respecting each vehicle's autonomy and the constraints due to the vehicles' features, revisits of sensible areas etc. etc. The following article describes a heuristic approach to the computing of these navigation plans.

Keywords: patrolling, optimization problem, path planning, submarine vehicles.

1. INTRODUCTION

The main activities carried out by the Coast Guard, are focused on improving the sailing security, the protection of the marine environment and ensuring the respect of National and International laws. For this purpose, a great number of resources are involved in daily coast patrolling operations, with consequently high costs. To reduce these costs, joint missions with other police teams are carried out. This is not always enough to ensure sufficient surveillance, due to, for example, a great number of coast's Kilometers (e.g. Italy has more than 8000 km of coasts). The risk is to leave some areas "unvisited", which might be particularly sensible, for which the maritime safety and/or the marine environment could be compromised. This article is focused on the problem of a submarine patrolling, or else the problem of patrolling sea sub-areas located at different depths. This is an important matter for what concerns the protection of marine environments and the improvement of the sea security.

Using "unmanned" vehicles to support patrolling operations would allow surveillance of great sea extensions with very few operators, allowing the local government, employed into monitoring of its territorial waters, to exploit in an efficient way its own resources of men and vehicles.

2. THE PATROLLING PROBLEM

2.1. A submarine sea area's patrolling

The problem of a submarine sea area's patrolling can be defined as follows: *"given a set of vehicles and an area, the paths assigned to each vehicle must be calculated taking into account that each point into the area would be visited by at least a vehicle"*.

Consequently, the following hypothesis can be formulated:

1. *Sailing autonomy*: each vehicle has its own sailing autonomy, which is the capacity of patrolling a certain amount of miles, depending on its fuel.
2. *Area points*: visiting some area points could be seen as visiting some interesting sites, such as buoys or strategic points, or else, given a precise sea area, visiting it all. We assume that vehicles have a 360 degrees sight of a certain ray, depending by the sensor installed on each vehicle.
3. *Sea Strength*: a point's sea strength is represented by an integer number between 0 and 9 which gives and information about the sea state into that area, based upon the Beaufort scale (i.e. 0=calm, 9=windstorm). Each vehicle has a certain sea capacity, defined by an integer number between 0 and 9 as well, that indicates the maximum sea strength that the hull can endure (i.e. a vehicle with sea capacity C can visit a point having sea strength F if $C \geq F$).
4. *Equipment*: each vehicle is equipped with particular tools and/or sensors. Some points of the area may require vehicles equipped with a certain tool (for example, it may be a weapon).
5. *Revisiting sensitive points*: some points of the area may be considered more "sensitive" than others, so that they require a revisit, for example at defined time intervals, by at least a vehicle.
6. *Obstacles presence*: some points might not be patrolled because of the presence of "obstacles" (i.e. islands, low backdrops, etc. etc...)
7. *Depth:* each vehicle can travel at different depths, considering its own depth constraint. Each node of the sea selected area is located at a different depth.

The optimization problem of the patrolling of a submarine sea area consists in searching for a "better" solution, composed by a set of paths, in order to visit the selected sea area fully, with the minimum time and considering the hypothesis above.
It's important to underline that the "best" solution (a more "efficient" patrolling) may involve only some available vehicles.

3. THE PATROLLING PROBLEM SEEN AS A MTSP

3.1. Variants to the original problem

Solution of the patrolling problem can be reconducted to the well-known mTSP in literature (multiple Travelling Salesman Problem) (see Brummit and Stentz (1996), Brummit and Stentz (1998), Yu et al. (2002), Ryan et al. (1998). The mTSP consists of a generalization of the Travelling Salesman Problem with more than one salesman (see Mole et al. (1983), Laporte et al. (1985), Toth and Vigo (2002)).
The classic mTSP formulation provides that the m salesmen must visit each city only one time, with the minimum possible "cost".
In the specific case of the patrolling problem, the following variants to the original formulation must be considered:

- *Multiple Deposits*: more deposits exist, with a certain number of salesmen dislocated into each of them. The salesmen can return into their own starting deposit after completing the tour or return into a random deposit (the initial number of salesmen must remain the same at the end of the trip).
- *Number of salesmen*: the number of salesmen can be represented by a limited variable or can be a fixed number.
- *Fixed cost*: if the number of salesmen isn't fixed, then each salesman has usually a fixed cost attributed, which has to be added to the function cost whenever this salesman is employed into the solution.
- *Time Windows*: some points must be visited into determined time intervals, named *time windows*. This is an important mTSP extension and it's named as Multiple Travelling Salesman Problem with Time Windows (mTSPTW) (see Macharis and Bontekoning (2004), Wang and Regan (2002), Ruland and Rodin (1997), Mitrovi´et al. (2004)).
- *Other restrictions*: these restrictions consist in a particular constraint on a particular equipment of the vehicle (salesman), which visits a point, on the vehicle's capacity to sustain sea's strength in that point, on the maximum length of paths attributed to each single vehicle, due to their autonomy and on the presences of obstacles.

4. PROBLEM DEFINITION

Given :

- A graph $G = (V,E)$ where V is a set of vertexes and E a set of arcs with a specific "cost" connecting vertexes;
- m, the number of salesmen (vehicles);
- deposits $D_i \in G$ from which salesmen must start their trip;
- $\mathbb{X}$ set of all possible configurations, or all possible choices of m paths starting from and ending into the assigned deposit and visiting once and only once each one of all the other vertexes;
- ν set of constraints;
- $\mathbb{X}_\nu$ sub-set of the configurations respecting the assigned constraints ν;
- $f : x \in \mathbb{X} \mapsto \mathbb{R}$ cost function assigned to the problem solution.

Solving the patrolling problem consists in finding a configuration $x \in \mathbb{X}$ which respects the constraints and minimizes the total cost, or: x=min(f(x)) with $x \in \mathbb{X}_\nu$.

4.1. Graph construction

The set V of graph's vertexes is built by coverage of the free-space by a Voronoi diagram. Cells of the diagram have a maximum ray compatible with the sensibility of the sensor used for patrolling, installed on each vehicle.

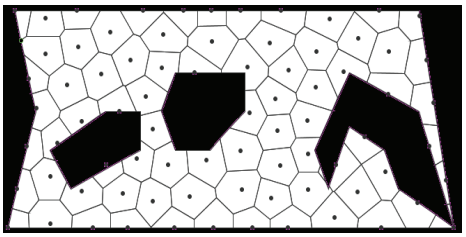

Figure 1: Voronoi Diagram

Vehicles and areas that have to be visited are considered as points while the obstacles are modeled as polygons, defined by their vertexes.
The set E is composed by all arcs of a complete graph built with the V set of vertexes, where each arc of the graph has an associated cost, equal to the Euclidean distance of its vertices, considering their latitude, longitude and depth.

4.2. Cost function

The cost function $f : \mathbb{X} \mapsto \mathbb{R}$ is defined on the set $\mathbb{X}$ of all possible solutions and is computed as:

$$f(x) = \sum_{p \in \{\text{paths of solution x}\}} [length\ (p) + penalty(p)]$$

with:

- *length (p)* = length of the path p (assigned to a vehicle) and belonging to the solution x
- *penalty(p)* = sum of all penalties of path p

penalties are computed as follows:

- *Penalty for vehicle's autonomy*

Autonomy is the maximum length of a path that a vehicle could make. The penalty for a path's autonomy p is:

$$penalty(p) = FMULT \cdot [c(p) - cmax(p)]$$

with:

- $c(p)$ = length of path p
- $cmax(p)$ = maximum length of p
- $FMULT$ = empirical multiplicative factor

The algorithm will try to satisfy the constraint considering high values of *FMULT* constant. But a too high value of *FMULT* may cause the algorithm to stop into a cost function's local minima.

- *Penalty for time windows*

Revisiting of one or more vehicles on a site is equivalent to define time windows (more or less regular) on this site, which at least a vehicle must visit. Time windows are attributed each one to a different node with the same position, added to the set V of graph G (i.e. if a node has to be visited twice, there could be added to the map other two nodes into the same position, but having different time windows). If a vertex v belonging to a path p is associated to a time window $[t1_v, t2_v]$ (with $t1_v$, $t2_v$ minimum and maximum visit time), and the visit time on that node, following p, is $t(v)$, the related penalty is computed as:

$$penalty(p) = TMULT \cdot [\sum_{v \in p} windowPenalty(v,t1_v,t2_v)^2]$$

with:

$windowPenalty\ (v,t1_v,t2_v) = 0$ *se* $t1_v \le t(v) \le t2_v$
$windowPenalty\ (v,t1_v,t2_v) = t(v) - t2_v$ *se* $t(v) > t2_v$
$windowPenalty\ (v,t1_v,t2_v) = t1_v - t(v)$ *se* $t(v) < t1_v$

and *TMULT* empirical multiplicative factor.

- *Sea Strength Penalty*

Sea strength on a vertex v is defined as an integer number $s(v)$ between 0 and 9, and each vehicle m sustain a maximum sea strength $s(m)$. A vehicle having strength $s(m)$ can visit a site with strength $s(v)$ if $s(m) \ge s(v)$.
Sea strength penalty of a path p is:

$$penalty(p)=MMULT \cdot (\#nodes\ v\ s.a.\ s(v) > s(m))$$

with *MMULT* empirical multiplicative factor.

- *Equipment Penalty*

Some nodes might have the constraint that they could be visited only by vehicles equipped with a particular sensor or tool (or weapon).
The equipment penalty of a vehicle with a path p is computed as follows:

$$penalty(p)=EMULT \cdot (\#nodes\ v\ s.a.\ m\ isn't\ provided\ with\ the\ right\ equipment\ required\ by\ v)$$

With *EMULT* empirical multiplicative factor.

- *Penalty for long arcs*

To avoid inserting long arcs into the final solution, optimizing the length of each sub-path, a possible choice is to associate a penalty if the distance between two connected nodes into a path p is longer than a certain length l, as *2*ray of the sensors* equipped on each vehicle.
The penalty for long arcs is as follows:

$$penalty(p)=LMULT \cdot (\#arcs\ e\ \text{associated to vehicle m}\ s.a.\ l(e) > 2*ray\ of\ the\ sensors\ \text{equipped on m})$$

With *LMULT* empirical multiplicative factor.

- *Obstacles Penalty*

A vehicle must necessarily avoid obstacles into the sea area that needs patrolling: this has been made inserting a certain penalty in case at least one of the arcs belonging to the set of paths would go through one of the obstacle's sides.

The obstacles penalty is computed as:

penalty(p)=OMULT if at least an arc belonging to the set of paths P goes through one of the obstacle's sides.

With *OMULT* empirical multiplicative factor.

5. HEURISTIC APPROACH

The heuristic approach used to solve the patrolling problem is the Simulated Annealing.
This algorithm is also called "meta-heuristic" and consists into an extension of the classical local search.
Considering the local search, when a solution's neighborhood is explored, the only information owned is the best current solution and the related cost function value.

5.1. Simulated Annealing

This approach, used to solve the patrolling problem, is inspired by the industrial process named annealing, and it's known in literature with the name of *Simulated Annealing* (see Aarts and Korst (1989), Dekkers and Aarts (1991), Romeijn and Smith (1994)). While a liquid's molecules tend to move freely at high temperatures, if a temperature is lowered in a sufficiently slow way, the molecules' thermic mobility is lost and they tend to form a pure crystal corresponding to a minimum energy state.

The annealing is a thermic treatment used mostly on steel and copper, the slower cooling we have, the stabler structure we obtain. Similarly, the approach tends to converge to an optimal solution (see Bélisl (1992), Locatelli (1996), Locatelli (2000)) using this heuristic and choosing a decreasing sequence of temperatures, with a sufficiently slow 'cooling': as we arrive to a minimum energy state through the physical process, in the same way we obtain a solution (the global optima) with a minimum cost function value, using the annealing into optimization problems.

The peculiarity of Simulated Annealing is the capacity to avoid local minima accepting also the transitions that increase the value of cost function f. Accepting configurations with a worse cost function is the only way to escape from local minima.

The heuristic is articulated into the following steps:

1. A sequence of temperatures $T_0>T_1>T_2>\ldots$ with T_i tending to 0 for $i\rightarrow\infty$ is fixed;
2. A rounded positive numbers sequence $N_0>N_1>N_2>\ldots$ and an iterator $j = 0$ are fixed;
3. The initial solution is generated;
4. $T = T_j$ and $N = N_j$, and an iterator $i = 0$ are set;
5. If $i <= N$ go to Step 6., otherwise go to Step 9.
6. A new random solution x' is generated (see par. 5.2)
7. If $f(x') < f(x)$ then $x = x'$, otherwise $x = x'$ with a certain probability:

$$p = e^{-[f(x') - f(x)] / T}$$

8. $i = i + 1$ and the heuristic returns to Step 5.
9. $j = j + 1$ and the heuristic returns to Step 4.

5.2. Random generation of a solution

To generate a solution randomly, the idea is to start from the last solution and choose randomly one of its "transformations" listed below:

1. Move 1-0 (Relocate)
2. Move 1-1
3. Move 2-0 (Double relocate)
4. Move 2-1
5. Move Or-Opt
6. Move CROSS
7. Move 2-Opt

Once a transformation to apply is found (*Move*), the vehicle containing the first vertex used for the exchange is chosen randomly. The other vertex (or others) involved into the exchange are chosen into the 'neighborhood' of the first vertex.

The different transformations used by the algorithm are:

1. *Move 1-0 (Relocate):* a vertex is moved into another position of the same vehicle's path, or else into another vehicle's path;

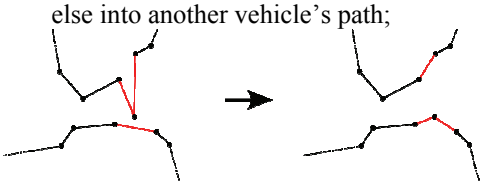

Figure 2: illustration of a "relocate" move

2. *Move 1-1:* a vertex into a vehicle's path is exchanged with a vertex contained into another vehicle's path;

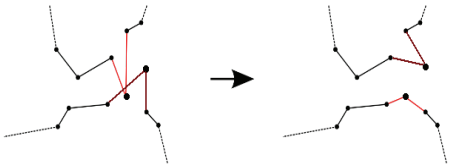

Figure 3: illustration of a move 1-1

3. *Move 2-0 (Double relocate*): a couple of near vertexes is moved into another position of the same vehicle's path or into another vehicle's path;

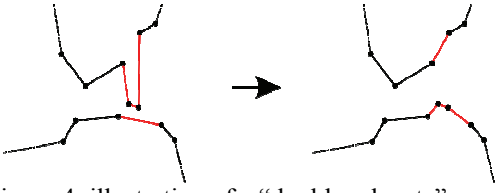

Figure 4: illustration of a "double relocate" move

4. *Move 2-1*: a couple of near vertexes is exchanged with a vertex contained into another vehicle's path;

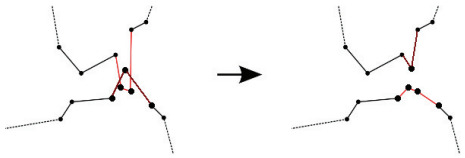

Figure 5: illustration of a move 2-1

5. *Move OrOpt:* two vertexes belonging to the same path swap their neighbors, maintaining all the rest unaltered;

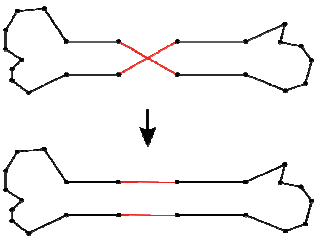

Figure 6: illustration of a move OrOpt

6. *Move CROSS:* two salesmen exchange a sub-path. This kind of move contains also the relocate ones and the exchanges between nodes of different paths (Move 1-1 and 2-1).

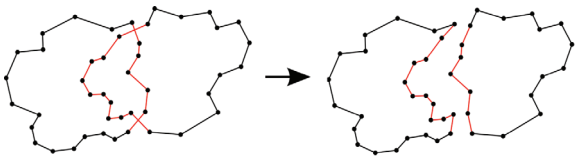

Figure 7: illustration of a move CROSS

7. *Move 2-Opt:* Two vehicles exchange their paths, starting from two vertexes. Two arcs get removed (i.e. *(i, j)* and *(h, k)*) and two new arcs are added, *(i, k)* and *(h, j)*. This is possible if and only if the two vehicles involved end their routes into the same deposit.

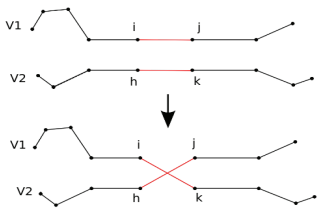

Figure 8: illustration of the move 2-Opt

6. EXPRIMENTAL RESULTS

Here below are shown some results taken from the application of this algorithm, using vehicles having the same maximum velocity but with different autonomy.

Each point of the area is defined by three coordinates:

- x = longitude
- y = latitude
- z = depth

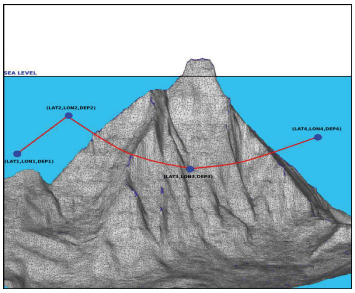

Figure 9: 3D representation of a path

The following cases have been considered:

1. case without any particular point into the sea area (so no particular constraints on sea strength, vehicle's equipment, nor revisits of sensible points)

2. case with all the constraints listed before.

6.1. Case without constraints

In *Figure 10* we may observe paths attributed to each vehicle into the selected submarine sea area: each *path* is represented by a set of arcs colored differently, each *node* is colored in grey, except for the *deposits*, each one having the same color of the path that starts from there. In this example, each node is located at a different depth.

Figure 10: Case without constraints 2D

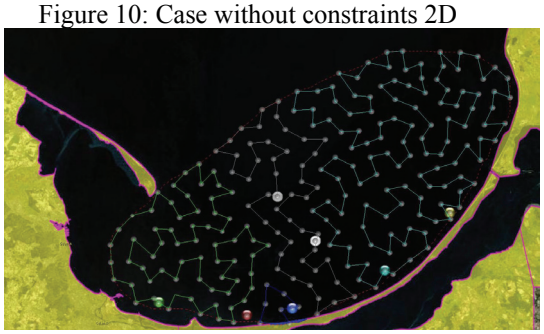

Figure 11: Case without constraints 3D

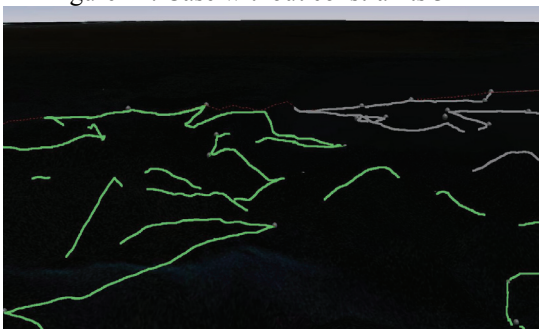

Table 1: Results without constraints

Vehicle	Used	Color	Autonomy (Km)	Vel. (Km/h)	Path Length (Km)
1	YES	RED	233,81	10	0,28
2	YES	BLUE	306,05	10	28,9
3	YES	CYAN	493,55	10	437,71
4	NO	YELLOW	86,91	10	0
5	NO	WHITE	166,3	10	0
6	YES	GREY	497,33	10	176,5
7	YES	GREEN	281,19	10	230,04

Total number of vehicles: 7

Vehicles involved: 5

Solution cost: 873.42Km

It's noticeable that the selected sea area is completely visited using only 5 among the 7 vehicles available.

6.2. Case with constraints

In Figure 11 we may observe paths attributed to each vehicle available into the selected submarine sea area, considering all the set of constraints described before: *nodes* without any particular constraint are represented in grey, nodes with *revisits* are colored in green, the ones with a certain *sea strength* are colored in red, while the ones needing a particular *equipment* are represented in yellow. In this example, each node is located at a different depth.

Figure 12: Case with constraints 2D

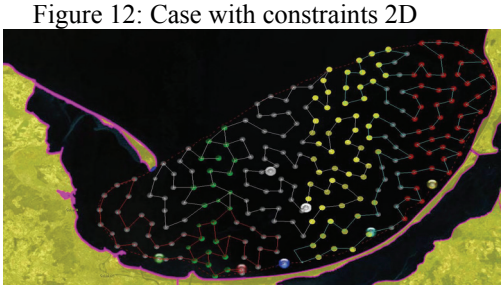

Figure 14: Case with constraints 3D

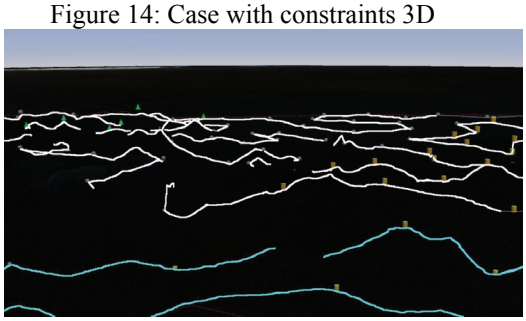

Table 2: Results with constraints

Vehicle	Used	Color	Autonomy (Km)	Vel. (Km/h)	Sea Strength	Equip.	Path Length (Km)
1	YES	RED	430	10	NO	NO	201,15
2	YES	BLUE	449,8	10	NO	YES	3,85
3	YES	CYAN	431,25	10	YES	YES	347,68
4	YES	YELLOW	176,43	10	YES	NO	0,34
5	YES	WHITE	467,92	10	NO	YES	407,15
6	NO	GREY	376,05	10	NO	YES	0
7	NO	GREEN	318	10	NO	NO	0

Total number of vehicles: 7

Vehicles involved: 5

Solution cost: 960.19Km

It's noticeable that the selected sea area is completely visited using 5 among the 7 vehicles available.

Vehicle 3 (in cyan), is the only one able to visit sites with high strength of the sea and sites that apply for special equipment.

REFERENCES

Aarts E.H.L., Korst J. (1989). Simulated Annealing and Boltzmann Machines. J. Wiley & Sons.

Bélisle C.J.P. (1992). Convergence theorems for a class of simulated annealing algorithms on Rd. Journal of Applied Probability, 29, pp. 885–892.

Brummit B, Stentz A. (1996). Dynamic mission planning for multiple mobile robots. Proceedings of the IEEE international conference on robotics and automation.

Brummit B, Stentz A. (1998). GRAMMPS: a generalized mission planner for multiple mobile robots. Proceedings of the IEEE international conference on robotics and automation.

Dekkers A., Aarts E., (1991). Global optimization and simulated annealing. Mathematical Programming, 50, pp. 367–393.

Laporte G, Nobert Y, Desrochers M. (1985). Optimal routing under capacity and distance restrictions. Operations Research, 33(5), pp. 1050–73.

Locatelli M., (1996). Convergence properties of simulated annealing for continuous global optimization. Journal of Applied Probability, 33, pp.1127–1140.

Locatelli, M. (2000). Simulated annealing algorithms for continuous global optimization: convergence conditions. Journal of Optimization Theory and Applications, 104, pp. 121–133.

Macharis C, Bontekoning YM. (2004). Opportunities for OR in intermodal freight transport research: a review. European Journal of Operational Research 153, pp. 400–16.

Mitrovi´c-Mini´c S, Krishnamurti R, Laporte G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. Transportation Research, 28(8), pp. 669–85.

Mole RH, Johnson DG, Wells K. (1983). Combinatorial analysis for route first-cluster second vehicle routing. Omega, 11(5), pp.507–12.

Romeijn H.E, Smith R.L. (1994). Simulated annealing for constrained global optimization. Journal of Global Optimization, 5(2), pp. 101–126.

Ruland KS, Rodin EY. (1997). The pickup and delivery problem. Computers and Mathematics with Applications, 33(12), pp. 1–13.

Ryan, J.L., Bailey, T.G., Moore, J.T., Carlton, W.B., (1998). Reactive Tabu search in unmanned aerial

reconnaissance simulations. Proceedings of the 1998 winter simulation conference, vol.1, pp. 873–9.

Toth P, Vigo D. (2002). Branch-and-bound algorithms for the capacitated VRP. In: Paolo Toth, Daniele Vigo, editors. The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, pp. 29–51.

Wang X, Regan AC. (2002). Local truckload pickup and delivery with hard time window constraints. Transportation Research Part B, 36, pp. 97–112.

Yu Z, Jinhai L, Guochang G, Rubo Z, Haiyan Y, (2002). An implementation of evolutionary computation for path planning of cooperative mobile robots. Proceedings of the fourth world congress on intelligent control and automation, vol. 3, p. 1798–802.