Mission Planning for Unmanned Aircraft with Genetic Algorithms

Hansen, Karl Damkjær

Publication date:
2014

Mission Planning for Unmanned Aircraft with Genetic Algorithms

Ph.D. Thesis

Karl Damkjær Hansen

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg

This document has been typeset in Latin Modern using the LaTeX system with the Aalborg University PhD thesis template of Jesper Kjær Nielsen. The graphics are produced with Inkscape and GIMP, and the photographs are produced by the research group itself or found at Flickr. The photographs from Flickr are licensed for use with Creative Commons Licenses; the specific license and name of the author is given in the image captions. The majority of scripting and programming of algorithms have been done in Python and C++.

For my wife Susan, without whom I would not have had a bad conscience for constantly thinking of helicopters instead of her for three years in a row. And for my unborn child, sorry I don't know your name yet. You can put it in here when you are old enough:

For _____, I love you.

Preface

The thesis in front of you is submitted for assessment as a plurality of papers in partial fulfillment of the requirements of the Doctor of Philosophy degree at the Section of Automation and Control, Department of Electronic Systems, Aalborg University, Denmark. This work has been carried out in the period from November 2010 to March 2014 under the supervision of Associate Professor Anders la Cour-Harbo.

I would like to acknowledge Anders for his support and supervision during this project, he has given me enough room to try several avenues of research, causing me considerable confusion and frustration, but also teaching me to structure and organize my work. At the right times, Anders have contributed with a genuine interest, enthusiasm, and work effort, I feel privileged to have had his support.

I would also like to acknowledge the entire ASETA team for pleasant company and serious collaboration both while sitting in our comfortable office chairs and while working long hours in the baking sun, raging wind, and pouring rain at the fields in Taastrup.

In the spring of 2013 I visited the Singapore-MIT Alliance for Research and Technology in Singapore. I would like to thank everybody there for a pleasant, fun, and learning experience, especially Professor Emilio Frazzoli for giving me the opportunity.

Last but definitely not least i would like to thank my wife Susan for mental support and listening attentively, even though she probably does not care about segmentation faults or transcendental functions.

Karl Damkjær Hansen
Aalborg University, March 12, 2014

Abstract

Unmanned aircraft invokes different feelings in people. Some see ruthless killing machines, other see a potential for fast and cheap distribution of goods, yet other see flexible and convenient emergency rescue drones. Regardless, advances and miniaturization in motors, sensors, and computer processing power have taken the unmanned aircraft from being a military application to the commercial sector and even into the hands of hobbyists.

Still, the enthusiastic interest in the new technology and its prospective advantages overshadows the fact that it mainly sees application where the aircraft are mostly under human command, just like remote controlled planes have been for years. Actually the *revolution of the drones* is not so much a revolution of the unmanned aircraft as it is a *digital control revolution*. Only a few years ago, hopeful remote-control pilots had to invest countless hours of training in mastering the planes, the controls were complex and originated from the stick-and-throttle controls of real fighter airplanes. Now, inherently unstable quad-copters can be controlled with touch screen, where a sliding motion to the right on the screen moves the aircraft to the right. Exactly such underlying automatic control methods has played a big role in popularizing the quad-copter as a toy, which in turn has awakened people's imagination and enthusiasm.

The next step of the unmanned aircraft is to become fully autonomous. Expert operators use unmanned aircraft to perform aerial surveys of nature conservation areas, construction sites, and the like. Although it flies autonomously, the operator need to understand the tool that the aircraft is to him. He must set the coordinates and altitude of each waypoint that the aircraft must visit, and he must decide on the sequence by which the waypoints must be visited. Surely, computer programs assist the operator in the planning process, but actually, the end product of the survey is not the flight plan of the aircraft or the aerial images that it takes, rather, it is the results of the analysis of the images; the politician or the entrepreneur who ordered the analysis probably did not care how the data was collected. Just like the control algorithms in the autopilot relieved the operator of the piloting burden, the next step will relieve him of the planning burden. The analyst simply defines which analysis she wants to perform and a plan is

automatically created for the aircraft, which collects the needed data in the best possible fashion. All she has to do is release the aircraft and collect it again when it lands.

In this dissertation we study the automatic mission planning for unmanned aircraft. The basis for the research is the case of agriculture automation where unmanned aircraft are used for aerial surveying of the crops. The farmer takes the role of the analyst above, who does not necessarily have any specific interest in remote controlled aircraft but needs the outcome of the survey. The recurring method in the study is the genetic algorithm; a flexible optimization framework that is used to perfect the flight plans.

Focus is given to planning under the kinematic constraints of the aircraft to obtain smooth trajectories that are much closer to a real flyable trajectory than the point-to-point waypoint trajectory. This focus results in the development of a method which models the aircraft as a Dubins vehicle and produces a plan that automatically decides on the headings and target speeds of a set of waypoints.

Another point of study is the constraint given by fuel limits. An aircraft can only visit so many waypoints before it must refuel. A method is developed, which plans for refueling stops in the sequence of waypoints, so that the unmanned aircraft can continuously survey a given area. This area is an important direction for research into long-term autonomy, where robots work for hours or days without human intervention.

Two more technical contributions are made in the area of the genetic algorithms. One is a method to decide on the right time to stop the computation of the plan, when the right balance is stricken between using the time planning and using the time flying. The other contribution is a characterization of the evolutionary operators used in the genetic algorithm. The result is a measure based on entropy to evaluate and control the diversity of the population of the genetic algorithm, which is an important factor its effectiveness.

Synopsis

Ubemandede fly påberåber sig forskellige følelser i folk. Nogle ser hensynsløse dræbermaskiner, nogle ser et potentiale for hurtig og billig distribution af varer, mens andre ser fleksible og bekvemme nødhjælpsdroner. Uanset hvad, har fremskridt og miniaturisering i motorer, sensorer og coputerkraft bragt ubemandede fly, fra at være en platform for militæret, til den kommercielle sektor og endda i hænderne på hobbyfolk.

Den entusiastiske interesse for den nye teknologi og dens potentielle fordele overskygger dog det faktum at den først og fremmest ser anvendelse i sammenhænge hvor flyet for det meste er under menneskelig kommando, ligesom fjernstyrede fly har været i årevis. Faktisk er der ikke sket et kvantespring i ubemandede fly, men nærmere et kvantespring i den digitale regulering teknik. For kun et par år siden måtte håbefulde fjernstyringspilotaspiranter investere utallige timers træning i at mestre flyene, kontrollerne var komplicerede og stammede fra de rigtige flys joystick og gashåndtag. Nu kan ustabile systemer som quad-copters styres med en touchscreen, hvor en glidende bevægelse til højre på skærmen bevæger flyet til højre. Præcis sådanne underliggende automatiske reguleringsmetoder har spillet en stor rolle i udbredelse af quad-coptere som legetøj, som igen har vakt folks fantasi og entusiasme.

Det næste trin for ubemandede fly er fuldstændig autonomi. Rådgivningsfirmaer bruger ubemandede fly til at udføre kortlægninger af naturbeskyttelsesområder, byggepladser og lignende. Men selv om det flyver autonomt, er operatøren nødt til at forstå det værktøj at flyet er. Han skal sætte koordinaterne og højde for hvert waypoint som flyet skal besøge, og han skal beslutte sig for den rækkefølge de skal besøges i. Operatøen har naturligvis computerprogrammer til at hjælpe i planlægningsprocessen, men faktisk er slutproduktet af undersøgelsen ikke flyets flyveplanen eller de luftfotos det tager, det er snarere resultatet af analysen af billederne. Politikeren eller entreprenøren der har bestilt analysen er sandsynligvis ligeglad med hvordan de pågældende data er indsamlet. Ligesom reguleringsalgoritmer i autopiloten har lettet operatøren for pilotarbejdet, vil det næste teknologiske skridt aflaste ham for planlægningen. En analytiker skal simpelthen kunne definere hvilken analyse hun ønsker at udføre, hvorefter en plan for flyet automatisk genereres, således at de nødvendige data bliver indsamlet på den

bedst mulige måde. Alt hvad hun skal gøre er at sende flyet afsted og samle det op igen når den lander.

I denne afhandling undersøger vi automatisk planlægning af missioner for ubemandede fly. Grundlaget for denne forskning er autmatisering af landbrugetet, hvor ubemandede fly anvendes til kortlægning af afgrøderne. Landmanden har som analytikeren ikke nødvendigvis har nogen særlig interesse i fjernstyrede fly men har brug for resultaterne fra analysen af billederne det tager. Den gennemgående metode i undersøgelsen er den genetiske algoritme, et fleksibelt optimeringsframework, som her bruges til at lægge flyveplanerne.

Et af fokuspunkterne er planlægning under flyets kinematiske begrænsninger for at opnå bløde baner der er meget tættere på reelle flyvebaner end punkt-til-punkt waypointbanerne. Denne fokus resulterer i en metode som modellerer fly som en Dubinsfartøj og lægger en plan hvor retningerne og hastighederne waypointene er automatisk bestemt.

Et andet punkt i undersøgelsen er den begrænsning som den medbragt brændstofmængde giver. Et fly kan kun besøge et vist antal waypoints før det skal tanke op. Det har resulteret i en metode som planlægger brændstofpåfyldning som en del af sekvensen af waypoints således at det ubemandede fly kan opretholde et konstant patruljemønster. Dette område er et vigtigt retning for forskning i langsigtede autonomi, hvor robotter arbejder i timer eller dage uden menneskelig indgriben.

To mere tekniske bidrag bringes indenfor genetiske algoritmer. Den ene er en metode til at træffe beslutning om det rigtige tidspunkt at stoppe beregningen af planen, hvor den rette balance er ramt mellem at bruge tid på planlægning og bruge tid på flyvning. Det andet bidrag er en karakteristik af de evolutionære operatører, der anvendes i den genetiske algoritme. Resultatet er en målemetode baseret på entropi til at evaluere og kontrollere mangfoldigheden af befolkningen i den genetiske algoritme, hvilket er en vigtig faktor for dens effektivitet.

Contents

Thesis Details

Thesis Title:	Mission Planning for Unmanned Aircraft with Genetic Algorithms
Ph.D. Student:	Karl Damkjær Hansen
Supervisors:	Assoc. Prof. Anders la Cour-Harbo, Aalborg University

The main body of this thesis consist of the following papers.

[A] Wajahat Kazmi, Morten Bisgaard, Francisco Garcia-Ruiz, Karl Damkjær Hansen, Anders la Cour-Harbo, "Adaptive Surveying and Early Treatment of Crops with a Team of Autonomous Vehicles," *ECMR'11*, 2011.

[B] Karl Damkjær Hansen, Francisco Garcia-Ruiz, Wajahat Kazmi, Morten Bisgaard, Anders la Cour-Harbo, Jesper Rasmussen, Hans Jørgen Andersen, "An Autonomous Robotic System for Mapping Weeds in Fields," *IAV'13*, 2013.

[C] Karl Damkjær Hansen, Anders la Cour-Harbo, "Characterization of Genetic Algorithm Mutation Operators for Solving Traveling Salesman's Problem," *Submitted for GECCO'14*, 2014.

[D] Karl Damkjær Hansen, Anders la Cour-Harbo, "Adaptive Termination Criterion for Genetic Algorithms Solving Traveling Salesman's Problems," *Submitted for GECCO'14*, 2014.

[E] Karl Damkjær Hansen, Anders la Cour-Harbo, "Aircraft Mission Planning with Refueling Constraints using Genetic Algorithms," *Submitted for IEEE IS'14*, 2014.

[F] Karl Damkjær Hansen, Anders la Cour-Harbo, "Waypoint planning with Dubins Curves using Genetic Algorithms," *Submitted for IEEE Trans. Autom. Control*, 2014.

This thesis has been submitted for assessment in partial fulfillment of the PhD degree. The thesis is based on the submitted or published scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extended summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The thesis is not in its present form acceptable for open publication but only in limited and closed circulation as copyright may not be ensured.

Part I

Introduction

Introduction

Every good report starts out with a compelling case. This report starts out with a story about a farmer.

The farmer is often made out to be the an evil, money hungry capitalist who depletes the earth of all its nutrient and poisons the environment, all for profit and personal gain. Most farmers would disagree with at least some of that description. In fact farmers tend to be pleasant people, not at all evil. It is important to observe that agriculture is indeed a business, and even though it may invoke feelings of being one with the nature or the like, farmers must make money. So money hungry may be an exaggeration, but without making money, any farmer would quickly be out of business. Because the farmer must plan to be in business for several years to come, there is no incentive for him to deplete all the nutrients of his fields one year to maximize the crop yield because he only has the same fields next year. On the other hand, the farmer must help his crops to keep his business profitable. One way is to fertilize the crops, to make sure that they have all the nutrients that they need to grow, another is to control pests such as weeds and insects, usually by means of spraying pesticides. This spraying is definitely one of the main reasons for the farmers' tarnished reputation. The fact is, however, that the money spent on pesticides is offset by the greater output of the crops. Actually, research show that weeds left uncontrolled may lead to a reduced crop output of 50 % or more [1].

Agriculture is a highly important old business, thus a lot of effort has gone into streamlining the workflow and development of machinery, pesticides, and fertilizers. These developments have enabled the farmer to singlehandedly run a farm with fields of more than 100 hectares. In Denmark, 62 % (2,639,905 hectares) of the country's land is used for agricultural purposes. This area is divided among about 40,000 holdings employing a total of about 40,000 full time farmers and about the same number of part-time employees [2]. About half of the holdings specialize in field crops, most of the other half specialize in animal production. In 2010, 3891 metric tonnes of pesticides were distributed in Denmark, 86 % of which was against weeds, the rest against fungi and insects. This equates to a total expenditure of 1.66 billion DKK annually. For the average crop producing farmer, the bill is 172.000 DKK.

The conventional way of distributing pesticides is via spraying booms mounted on tractors. This distribution is very elaborate, much more than one might think. To achieve an effectiveness of the pesticides, they must be applied to the leaves of the plants rather than the ground. This has lead to several technical innovations: The spraying nozzles have been perfected to produce vortexes that distribute the pesticides mostly onto the leaves. The booms have been lowered to hover just above the crops to avoid the wind carrying the aerosols away. This has lead to the development of boom stabilizing equipment, so that the height of the boom is maintained when the tractor hits a rock, a hole, or the like. These developments work to increase the effectiveness of the pesticides, while another category of developments aims to relieve the manual work of the farmer. This category seems build on the premise that bigger is better. One development is larger booms to spray larger areas, reducing the time that the farmer uses in the field. Another is larger tanks to carry more pesticides which reduces overhead when refilling the tanks. This requires larger tractors to pull the larger tanks. In the end, all this is to reduce the manual labor spent per hectare, because the economic result of the farm is usually only just enough to keep up one or two full time jobs.

Fig. 1: Preemergence spraying on rapeseed. This boom is 40 meters tip-to-tip. *CC-BY-2.0 Chafer Machinery*

Unfortunately, the bigger-is-better philosophy has some drawbacks. One of which is the compaction of the soil due to the heavy machines. This compaction inhibits the growth of the roots, making for a smaller yield. In turn, the compaction turns out to play a large, although hidden, factor in the fuel consumption of agriculture, because the farmer needs to till the soil to loosen it up. Tillage is a very fuel consuming operation. To put it into perspective, it roughly equates to dropping an anchor into the ground and pulling it around. The compaction further contributes to the fuel usage as the soil becomes harder to till.

In the longer run, compaction may significantly deteriorate the soil. The topsoil is loosened by the tilling but the subsoil is not, and thus the compaction is cumulative. The compacted subsoil acts as a sealing, reducing water, gases, and nutrient flux in the

field, leading to less biodiversity. Research deems compaction one of the key threats to the sustainability of soil in the European Union [3], with a potential high impact on production.

Fig. 2: Farmer tilling his field. *CC-BY-2.0 Alfonso Benayas*

Automation through autonomy may hold the solution to both reducing the pesticide usage and avoiding soil compaction. The fundamental premise for both these problems; minimizing the time of manual labor per hectare, is still in focus. However, the constraint that the farmer must use his time in the cockpit of his tractor is removed. Autonomous tractors may roam the fields while the farmer tends to other more advanced tasks. When the farmer is removed from the equation, it is not necessary either, to think of the tractor in the conventional way. Maybe instead of one big tractor, tens or hundreds of smaller, lighter, and more agile vehicles can be used, each carrying a smaller implements.

The small autonomous fleet also holds the promise of *precision farming*, the solution to over-usage of pesticides and over-fertilization. The small vehicles may autonomously measure the state of every small patch of the field, and treat it according to the individual needs. Traditionally, the treatment of the fields is quite open looped without much feedback. Farmers make general observations of their fields and treats them according to those observations, along with recommendations from models, and actual experience. Naturally, the farmer does not spray unnecessarily as the chemicals constitute a large item in his financial accounts, but loosing his crops due to pests is even worse. So the spraying tend to be performed in a better-safe-than-sorry fashion. The small autonomous vehicles can help in this case. Instead of general observations, the vehicles can be equipped with a comprehensive sensor suite, enabling them to measure the soil and the plants and differentiate the treatment for the different parts of the field. This leads to a vision of a system of autonomous vehicles roaming the fields, gathering site-specific information about small patches of soil or even individual plants while col-

laborating with other vehicles, treating the field according to that specific information while saving the farmer money by reducing his manual workload and usage of pesticides and fertilizers.

The ASETA project envisions such a system, using not only ground-based vehicles but also aerial vehicles. These unmanned aircraft will provide quick overviews of the fields to target the information collection to sites that look problematic, so that the ground-based vehicles will not blindly collect information for the entire field but quickly go to areas of high interest.

This thesis will continue to describe the proposed ASETA system, then delve into the specifics of planning the flight of the unmanned aircraft in that system. The recurring theme will be the use of genetic algorithms for this planning.

1 The ASETA System

Adaptive Surveying and Early Treatment of crops with a Team of Autonomous vehicles (ASETA) is a collaborative project between Aalborg University, University of Copenhagen, and Nordic Beet Research to construct a prototype system of autonomous systems for the agriculture composed of both unmanned aircraft and ground vehicles. The future commercial off-the-shelf system that the research in the ASETA project will contribute to is an expert system that autonomously collects information about a farmer's fields, generates treatment plans, and executes these if the farmer approves of them. Consequently, the farmer is relieved of much trivial work, while still being in charge of the production on his holding.

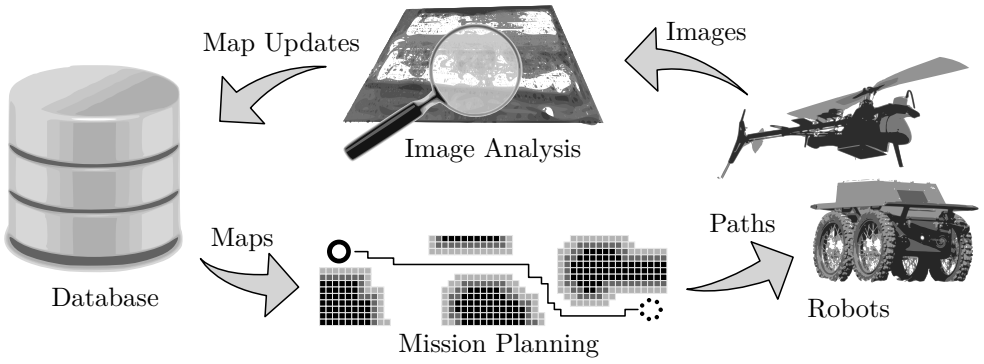


Fig. 3: Concept of the ASETA system. Hotspots are identified from images captured by the robots and stored in a database. If the database contains unsurveyed areas, the mission planner dispatches the robots to collect images. The process continues until all of the field is satisfactorily surveyed and all hotspots identified.

The strategy is a multi-level approach, which employs the unmanned aircraft to perform an initial survey of the field from high altitude, then improve the survey from lower altitude, and finally perform close-to-crop measurements with the ground vehicle. First, the unmanned aircraft identifies *hotspots*, which are areas that are identified to be off par with the expected state of the field. As the aerial images are taken at a very high altitude, the resolution may not be sufficiently good to draw satisfactorily conclusions, so the areas deemed to look problematic are scheduled for a fly-by at a lower altitude to obtain clearer images. When the better-resolution images have been analyzed, the ground vehicles are dispatched to obtain samples from within the hotspots. As the system sift through the field and sufficient information have been collected an appropriate treatment is planned.

Because the autonomous vehicles are not subject to the same time constraints as the human farmer, more time can be used per hectare, wherefore more attention can be given to the individual plants. The long term vision is for the autonomous vehicles to treat each plant individually; enabling selective pesticide application so that only weed plants are sprayed. The savings from individual spraying are hard to estimate, but have been estimated to potentially reduce the pesticide usage to a figure between 50 % and 95 %. But the individual weed control is not limited to only pesticide application, mechanical weeding may also be an option just as research has been done in laser treatment [4].

Fig. 4: The autonomous helicopter from the ASETA project.

The aircraft used in the ASETA project is an autonomous helicopter based on a Maxi Joker 3 remote controlled electric helicopter. The helicopter is equipped with a guidance, navigation, and control system developed at the UAS laboratory at Aalborg University. The ground vehicle is based on a four-wheeled skid steered RobuROC4 from Robosoft. Both vehicles are equipped with high-precision RTK GPS sensors and inertial measurement systems. The vehicles are fitted with different camera equipment. The helicopter carries a multi-spectral camera that captures images in six narrow bands from

near-infrared through the visible spectrum to blue, that enables image analysis on the aerial images to differentiate between weeds and crops. The ground vehicle carries a combination of a standard industrial RGB-camera aligned with a time-of-flight (TOF) camera, which, instead of measuring colors, measures distances enabling the analysis to detect outlines of leaves quite easily; a task that is quite hard to accomplish with regular color images.

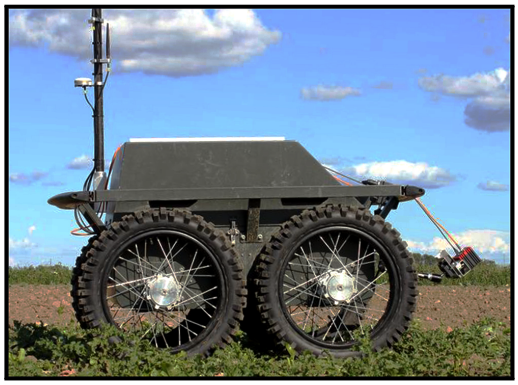


Fig. 5: The RobuROC4 autonomous ground vehicle from the ASETA project.

ASETA works with a case of mapping and removing thistles from sugar beet fields. This case was chosen at the beginning of the project because of it being relatively approachable. The leaves of thistles and sugar beets are fairly different with regards to shape and color, so it was expected that it would be possible to develop methods for automatic identification of the different species, which was indeed the case.

The aerial image analysis is based on the color difference between the plants. The six light bands of the multi-spectral camera were selected in the ranges where the differences in colors were most pronounced. At high altitudes however, each plant is not uniquely identified as each pixel catches light from more than a single leaf. These images provides the basis for a hotspot analysis; the more an area seems to be influenced by the thistle color notes, the higher the weed-to-crop ratio is. This analysis does not provide us with an absolute measure of the weed density, but rather a relative measure. Because of this, closer inspections of the hotspots are needed. As the first measurements are relative, only few close inspections are needed to calibrate the values; providing information of other hotspots with the same relative values.

Aerial images taken at lower altitudes will permit discrimination between individual plants, and while color analysis is still valid, other methods also become possible. One such method is to take advantage of the fact that sugar beets are planted in structured rows with rather fixed intervals. Thus any plant in between the rows is not wanted and may be flagged as removable. In the same manner, the average density of the rows

may be characterized and any deviation from this recorded, so that higher density may indicate a weed plant located in the row and lower density may indicate problems with nutrition or the like.

Fig. 6: The test field used for the ASETA project at the university farms of University of Copenhagen's Faculty of Life Sciences.

Just as the lower altitude aerial images, the ground vehicle can also provide an absolute measure of the weed density. The same general methodology may be applied; finding weeds in between the rows and detecting row infiltrations. Additionally, the ground vehicle has the advantage of being so close to the crops that it can log the absolute location of the plant with very high precision; something that the helicopter has trouble with due to wind gust, camera misalignments, and so on. Further the ground vehicle can identify the type and growth stage of the plants by means of leaf size, shape and count.

Paper A presents a further explanation about the equipment and vehicles used in the ASETA project, and Paper B presents a more in-depth treatment of the ASETA task management system.

The conceptual flow of data is shown in figure 3, and the process of turning images into maps have been outlined above. The last step in completing the circle is the automatic mission planning. In this step, missions are constructed for the robots, so that they collect the most relevant information for the system and treat the fields accordingly, all done in the shortest time and with the least waste of resources. Such planning requires the use of algorithms from the areas of operations research and artificial intelligence to determine the division of tasks between the individual robots. This system may incorporate extensive models for the fields, include weather forecasts, and historical data to plan missions that results in sensible treatments, yielding outputs that are optimized for the long run, not just the current season, all while conforming to national and international regulations.

The mission planner must work on different level with very different timespans. Planning the treatment of the fields is a slow process, where large amounts of data needs to be collected and processed before a conclusion can be drawn. In the other end of the scale is the path planning for the robots to make sure that the robots go where they are needed. This process works in the seconds and minutes range; deciding on which robot should go where, and how and when it should happen.

The rest of this thesis will investigate aspects of the path planning for the ASETA system, with a focus on planning for the helicopters.

2 Robotic Path Planning

Path planning is the process of constructing a path that takes a mobile robot from one position to another. This is a large area of research as there are many levels of planning and many approaches.

Starting at the very basic level is point-to-point planning; how a robot located at a point in space is guided to a given goal-point. A control law to guide the robot towards the goal may be a simple proportional controller that drives the heading and speed of the robot towards the point. If the robot is located in a obstacle-free and completely homogeneous environment, there may not be much more involved in the point-to-point planning. In the presence of obstacles however, things become more tricky. Navigating around these obstacles is one of the planner's finest tasks. A popular method to overcome this challenge is to partition the environment into a costmap; a 2D (or 3D) discretization of the environment, with each bin holding a value corresponding to how close it is to an obstacle. The costmap can be translated into a graph with the nodes corresponding to the coordinates of the bins, where the nodes are connected if the bins are adjacent. The task is then to find the sequence of steps between adjacent bins that joins the initial and goal point which minimizes the cumulated cost. Several methods have been developed to find the minimal cost path through such a graph, one is the A* algorithm by Hart et al. [5],which extends the well-known Dijkstra's algorithm with a heuristic to guide the search towards the goal. Finding the lowest cost sequence ensures that the robot does not come into conflict with the obstacles and keeps a sufficient distance. The found sequence can now be used with the original point-to-point control law by feeding the robot with a new point whenever the robot comes close enough to the current target.

Another concern of the planner is to produce paths that is in fact feasible for the robot to follow. The robot has a set of kinematic constraints that limits its movements. For example, a car-like Ackermann-steered vehicle has a turning radius corresponding to the maximal steering angle of the front wheels. Such constraints may significantly increase the cumulated cost of the above mentioned plan as the robot will deviate from the course. In the worst case, the path may be infeasible, and the robot is not even be able to negotiate the turns to avoid the obstacles. One of the chief reasons for

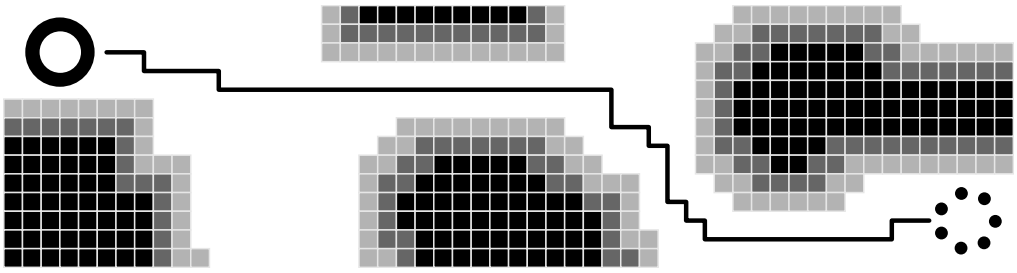


Fig. 7: Costmap with a path from the starting position (solid circle) to the goal position (dotted circle).

planning with the kinematics is to obtain robust behavior that does not require human supervision.

Another set of constraints that may interfere with the quality of the executed plan is the dynamics of the robot, i.e. the constraints set by the motion rather than the structure of the robot. Examples of dynamic constraints may be the breaking distance of the previously mentioned car, which is longer for higher speeds, or tire slippage, which results in wider turns at higher speeds.

When planning under dynamic constraints, the result is often not only a path in space but also in time, such that every point in the path has both a physical coordinate, a target time and potentially a velocity vector, acceleration vector, etc. Such a path is usually called a trajectory.

2.1 Case: Kinematics of the Helicopter

The helicopter is an interesting aircraft. One of its most prominent capabilities is its ability to perform vertical liftoff and landing, another is to hover. With miniaturization, the usefulness of small scale helicopters has greatly evolved. Recently, it has seen enormous use as camera platforms in both professional and hobby film production as well as surveying drones in civil and military applications.

Much work is being done in the so called D3 areas (Dull, Dirty and Dangerous) to aid and replace human operators in harmful situations such as firefighting, disaster management and rescue operations. The small-scale helicopter may traverse terrain that wheeled or tracked robots are unable to, it may enter holes in walls, and because of its low cost it can be dispatched into dangerous areas from where it may not return.

In conventional air travel the prevalent helicopter type has one main vertical rotor and one horizontal tail rotor. In small scale helicopters, the types are much more diverse. A popular category is the multi-rotors, which are usually fitted with fixed pitch rotor blades mounted directly on the rotor shaft of electric motors mounted symmetrically around the center where the battery and electronics are mounted. Of these types,

the quad-copter (with four rotors) is extremely popular with hobbyists and in research in swarm and indoor flight. Other types include the usual main-rotor and tail-rotor configuration, the tandem main rotor configuration, where the rotors spin in opposite directions, and the ducted fan configuration, which has a single main rotor and down facing control vanes. The kinematics of all the helicopters types are largely alike, here we will have a look at the traditional main-rotor tail-rotor configuration.

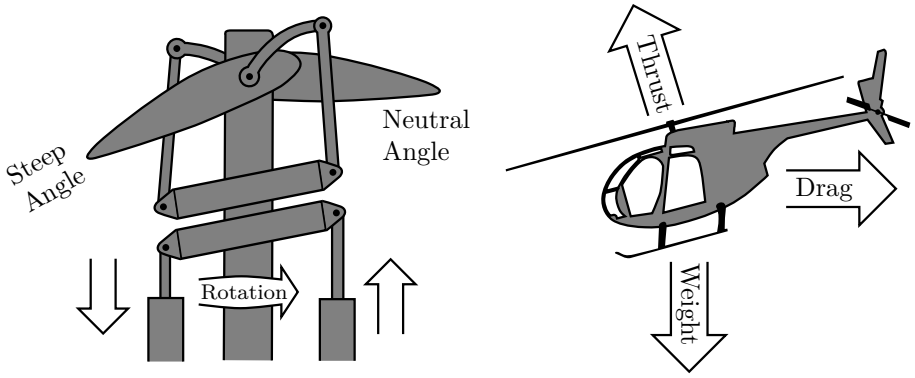


Fig. 8: Left: Conceptual drawing of a swashplate linkage, which controls the pitch of the rotor blades. Right: Free body diagram of a helicopter in forward flight.

The main rotor blades are individually hinged and coupled via a swashplate to the control actuators. This allows control over the pitch of the blades, so that thrust may be varied both collectively to force the helicopter upwards or downwards and cyclically to pitch the nose up or down and roll right or left.

The free body diagram of the helicopter in figure 8 shows how the main rotor can provide forward trust when the body is pitched down, likewise the body may be rolled to the side to move laterally. This analysis shows how a helicopter moving forward at a constant speed will exhibit a turning radius as a maximum banking maneuver can only provide a finite centripetal force, leading to a circular movement. Indeed, this is the same case for fixed wing aircraft.

Path planning with a comprehensive model for the helicopter is quite computationally intensive and quite possibly not necessary, since inherent inaccuracies in sensors and actuators as well as in-flight disturbances from wind means that the helicopter cannot track a trajectory completely accurately anyway. Instead the model can be reduced to a suiting simple model that provides enough likeness with an actual helicopter, at least in the normal operating conditions like forward flight and hovering. Instead of planning with the reduced model, another approach is a two-step process of first finding a plan with an even simpler cost function, such as the Euclidean distance so the plan can be found with e.g. the A* algorithm as described before. Next, the model is used to smooth

the path so that it is traversable by the reduced model. This usually includes rounding the sharp corners introduced by the A* that the kinematics of the model cannot accommodate. The cost of the smoothed plan can then computed from the costmap to make sure that the path does not collide with objects.

How the solutions should be smoothed, depends on the controller of the robot. The Pure Pursuit control strategy [6] is derived from fighter plane dog-fighting tactics, where the nose of the fighter plane is pointed towards the plane that it is tracking. This control strategy uses the straight lines between the points of the A* solution, projects the robots position onto this line, and sets a goal point for the robot on the line offset a measure towards the actual goal; the endpoint of the line. This way the robot is guided along the straight line segments, like a pull cart, converging to the line. The downside of this approach is that there is no guarantees that the robot will avoid the obstacles in its path. Therefore a simple model may be propagated through the solution with a Pure Pursuit controller to get some degree of assurance that the plan is feasible.

One model reduction of the helicopter is the Dubins vehicle. By connecting straight line segments and circle segments any two configurations consisting of a two-dimensional coordinate and a heading may be connected, see figure 9. L.E. Dubins showed that a combination of three circle segments (CCC), two circle and a straight line segment (CSC), or a subset thereof is sufficient to connect any two configurations and that this path is the shortest attainable [7] given the curvature constraint that defines the circle. The Dubins curves are often used in robotic path planning because they accurately match the kinematics of Ackermann-vehicles, although they ignore the dynamics of the vehicle originating from its inertia and the turning time of the wheels. Shkel and Lumelsky formulates convenient formulas for the Dubins curves [8] along with a classification scheme to determine which of the six configurations is the shorter. Otherwise, an often used method is to simply compute and compare all the six curves.

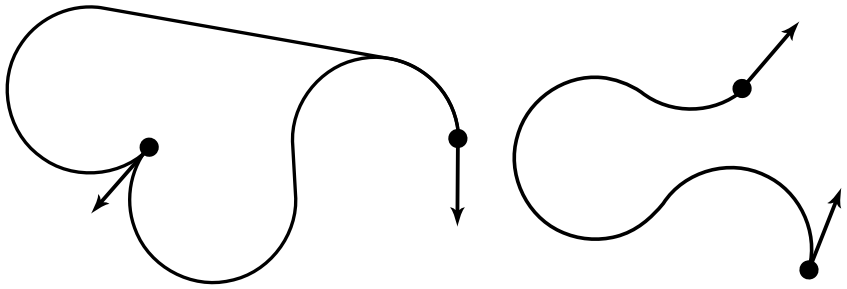


Fig. 9: Examples of Dubins curves. Left, a Right-Straight-Right combined with a Left-Straight-Right curve and right, a Left-Right-Left curve.

An example of a smoothing method of the Euclidean solution with the Dubins vehicle is the Alternating Algorithm (AA) by Savla, Frazzoli, and Bullo [9]. This algorithm pairs

every odd numbered node with the following node and sets the headings of those nodes to point in the same direction, the first towards the other and the other away from the first. This way the shortest Dubins path is the straight line. Each of the straight segments are then joined with a fitting Dubins path. Figure 10 shows an example of a Euclidean solution to a TSP smoothed with the Alternating Algorithm.

A Dubins path adds the constraint that the joined segments are co-directional, thus the path has C^1 continuity but a non-continuous second derivative because the curvature exhibits a step where the sections are connected. In reality, it is not possible to follow such a path in a vehicle moving at constant forward speed as the angular acceleration is infinite. This path inaccuracy may be a problem in some applications, and a C^2 continuous path is needed. In that case, the clothoid or Pythagorean hodograph may be solutions. Both of these along with Dubins paths are described in [10] along with other interesting path planning concepts.

In Paper F the traditional Dubins path is extended to include both variable radii and variable forward speed.

2.2 The Traveling Salesman Problem

Until now we have mainly focused on the point-to-point planning, where the objective is to move from an initial configuration to some goal configuration. On top of this we can add another layer of planning; mission planning. In the ASETA project, the helicopter is tasked to photograph the field from a set of waypoints. The database system has no preference as to when the different pictures are taken or in which order. So the mission planner for the helicopter is free to choose the sequence by which the waypoints are visited.

The general formulation of such a problem is called the traveling salesman problem (TSP). It stems from the problem a traveling salesman is faced with when deciding on the shortest route between a number of destination cities, where the route must end in his own home town where he set out from. This problem is surprisingly difficult. The number of possible solutions is

$$N_s = \frac{(N_c - 1)!}{2}$$

where N_c is the number of cities to visit. Because the direction of travel is unimportant, the solution space is halved, but even so, for just moderate sized problems, the solution space is enormous.

The problem seems related to the shortest path problem described in the point-to-point planning in costmaps, although here the problem is to visit all the points in the graph, whereas only the subset needed to produce the shortest path was needed before. This fact changes the problem from moderately hard to very hard. In fact, the problem was characterized in the classical paper by R.M. Karp on 21 combinatorial problems [11]. Karp showed that these problems were $\mathcal{NP}$-complete, which means that

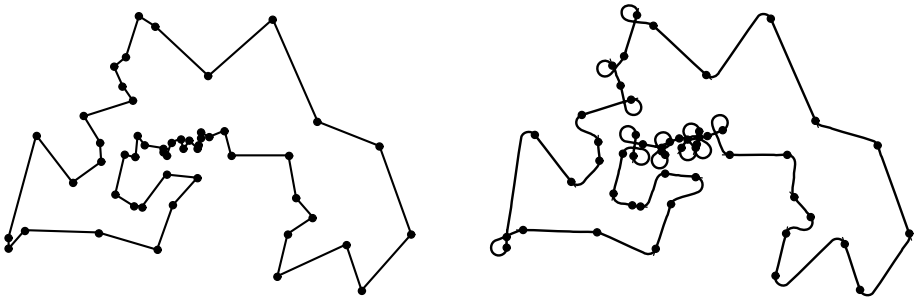


Fig. 10: Euclidean solution to a 52-city problem in its Euclidean form (left) and smoothed by the Alternating Algorithm (right).

the worst-case running time of a solving algorithm will grow super-polynomially with the problem size. Among Karp's 21 problems is the Hamiltonian cycle problem, that is the problem of finding a path in a graph which visits each vertice exactly once. The TSP is a specialization of the Hamiltonian cycle problem with the constraint that the edges of the graph are weighted, representing the distance between the cities, and the solution must be the Hamiltonian cycle with the least cumulated cost. This implies the $\mathcal{NP}$-completeness of the TSP.

Since its first mathematical treatment in the 1930s by K. Menger [12], several researchers have turned their attention to this problem. A notable approach is that of Dantzig, Fulkerson, and Johnson [13], who formulate the problem as an integer programming problem and solves it with a cutting-plane method. This algorithm has later been used by Applegate et al. in the software package Concorde [14], which has been used to solve problems of size in the tens of thousands.

Another approach is the heuristic approach of Lin and Kernighan, they build upon a local search method called n-opt. This method takes n nodes in the TSP graph, evaluates all the possible permutations, and chooses the one with the lowest cost, this procedure is repeated for all combinations of n nodes. The heuristics 2- and 3-opt are popular for obtaining fast and fairly good solutions to TSPs. Lin and Kernighan developed a method which uses opt moves of variable degree [15]. This algorithm has been implemented by K. Helsgaun in the LKH package [16], which competes with Concorde to solve the largest TSP instances.

In the lines of the Lin-Kernighan heuristic approach, meta-heuristics have also been used to solve the TSP problem. The name meta-heuristic refers to the fact that they are not readily implementable to solve specific problems like the Lin-Kernighan is, rather they provide a general structure of algorithms that have been shown to work well in a number of optimization domains. Of these meta-heuristics, a notable couple of methods are hill-climbing, simulated annealing, ant colony optimization, and genetic algorithms.

The different types of meta-heuristics differs slightly in requirements, but for them to work, they usually need an implementation of a problem specific neighborhood function like the n-opt move, which transforms one possible solution into a *neighboring* solution. Furthermore, they usually need a termination criterion, that stops the computation when the algorithm seems to have found the best solution.

Most of the meta-heuristics are based on processes that seem to optimize some aspect in the real world. It is the hope that the optimization in that domain may be applicable in other domains. This feature also serves as effective mnemonics for remembering the structure of the algorithms.

The hill-climbing procedure simply chooses the newly generated solution from the neighborhood function if it is better than the current solution, analogously to always climbing upwards. This usually leads to the algorithm getting caught in local optima, thus the initial guess on a solution must be sufficiently close to the optimal in order to obtain that solution.

The simulated annealing algorithm by Kirkpatrick et al. [17] solves this problem of getting caught in local optima by allowing a measure of random behavior when choosing the next solution. This way the algorithm may escape the local optima. The method derives from the controlled cooling of heated metal to improve its homogeneity. This happens because of thermal fluctuations in the metal, which are determined by the Boltzmann distribution. At higher temperatures, the atoms in the metal fluctuates wildly, but as the temperature is lowered, they settle down into ordered structures. In simulated annealing, the hill-climbing is controlled by a cooling plan. Initially, the algorithm is allowed to accept solutions that moves the algorithm far down the hill; corresponding to thermal fluctuations, but as the temperature approaches zero, the fluctuations cease and the pure hill-climbing behavior remains.

A rather different approach is the ant colony optimization proposed by M. Dorigo [18]. This method is based on the behavior of scouting ants, who lay out pheromones when searching for food for the colony. The worker ants may then follow these pheromones to the depots of food. This method is intuitive for graph search applications, as *pheromones* are laid out on the edges in the graph when being a part of a solution. After several runs of the algorithm, more and more pheromones are laid out, and the edges with most pheromones will define the best tour.

Another approach again is the genetic algorithm, which is based on the process of Darwinian evolution. It maintains a population of prospective solutions and via a *survival of the fittest* selection, the solutions are *mutated* via the neighborhood function. Furthermore, the chosen solutions may be combined in a *crossover* function. The selection process is based on probability in a roulette-wheel fashion, such that the most fit solution has the highest probability of being chosen while the least fit also has a slight chance. The randomness and diversity of the population serves the same purpose as the thermal fluctuations in simulated annealing to escape from local optima. The idea of Darwinian evolution is to randomly mutate the genomes of the most fit solutions in

order to breed even more fit solutions as generations pass. This does not mean that less fit offspring is not produced however, but that the evolutionary pressure of selection works to reduce their reproduction. A very basic genetic algorithm is shown in algorithm 1. This algorithm can be used with different crossover and mutation operators and extended with extra features, some of which will be described later.

Algorithm 1 A basic genetic algorithm.

Require: N_p
  procedure GENETIC ALGORITHM
    Initialize *population* $\leftarrow N_p$ random genomes
    $doTerminate \leftarrow False$
    while not $doTerminate$ **do**
      $newPopulation \leftarrow$ empty population
      Make ROULETTEWHEEL selector from *population*
      for $i \leftarrow 1, N_p$ **do**
        $parent_1 \leftarrow$ ROULETTEWHEEL
        $parent_2 \leftarrow$ ROULETTEWHEEL
        $child \leftarrow$ CROSSOVER$(parent_1, parent_2)$
        $child \leftarrow$ MUTATE$(child)$
        Add *child* to *newPopulation*
      end for
      population $\leftarrow newPopulation$
      $doTerminate \leftarrow$ evaluate termination criterion
    end while
  end procedure

One of the interesting features of these methods is that they usually exhibit what is called anytime behavior; that the algorithm may be stopped at any time during computation, and it will be able to present a feasible, although not necessarily optimal solution. This is an interesting feature in applications with deadlines.

For a comprehensive introduction to meta-heuristics, a good reference is the *Springer Handbook of Meta-Heuristics* [19]. Other excellent textbooks in the area of robotic planning include *Planning Algorithms* by S. M. LaValle [20], *Probabilistic Robotics* by S. Thrun [21], and *Principles of Robot Motion* by H. Choset et al. [22].

3 Genetic Algorithms for Traveling Salesman Problems

Genetic algorithms have received a great deal of attention in the context of TSPs. We will have a look at the general structure of this meta-heuristic and continue to look at the specific implementations needed to use it for solving TSPs.

The definition of the genetic algorithm is usually attributed to J. H. Holland [23] for his description of the algorithm with parallels to the mechanics of Darwinian evolution. In the genetic algorithms, each possible solution to the problem at hand is defined by its genome. Just as the genomes in humans is not a direct representation of the human but rather an encoded blueprint, the genomes of the genetic algorithm need not represent the solutions in any obvious way. Rather, it may be transformed into its final form by a transformation function. The encoded form of the solution is called its genotype and the final form is called its phenotype. When the solutions are mutated in the algorithm, it is the genotype that is transformed rather than the phenotype.

One reason for the division between geno- and phenotype is the possibility to represent the phenotypes compactly, such that the mutations transforms the solutions orthogonally. Holland formalized this using a schema analysis. He defined the genomes as binary strings so that a possible schemata may be $\{0\ 1\ *\ *\ 1\ *\}$ which represents fixed values in first, second, and fifth position and $*$s represent wildcards. Such a schemata should then represent a *cutting plane* in the solution space. By finding the optimal solution within this schemata, the global optimal solution should conceptually exist in the schemata based on that local optimum with wildcards in three formerly fixed positions. This orthogonal property is not necessarily attainable as we shall see later.

Another approach to evolution is based on the ideas of Jean-Babtiste Lamarck, who in the early 19^{th} century preceded Darwin with a possible explanation of evolution. Lamarck believed that individuals adapt to their environment by acquiring new traits and knowledge, and that these are passed on to the offspring, whereby the new generation attains a higher degree of fitness to the environment. The advances in molecular biology with the discovery of DNA seem to confirm that the Darwinian model is indeed the model at play in nature. That fact, however, does not limit the application of Lamarckian evolution in genetic algorithms. Such a scheme may be implemented alongside the Darwinian evolution to form a hybrid evolution by performing a local optimization on each solution in the population at every generation. The changes from the local optimization are then written back into the genome and the algorithm proceeds to perform selection in this optimized population.

The applicability of the genetic algorithm somewhat depends on the structure of the solution space that it is supposed to search. The space can be conceptually thought of as a landscape where the top of the hills are good solutions and valleys are bad solutions. See figure 11. The genetic algorithm is based on a population of candidate solutions. This feature is particular useful in the case of *hilly* search spaces in that the

different candidates need not explore the same hill for the optimal solution but may be divided between the individual hills. This feature has some resemblance to the mechanics of branch-and-bound algorithms as the individuals exploring the smaller hills tend to die out when other individuals find taller hills; increasing their fitness and chances of reproducing. If the space has too many hills compared to the size of the population, the algorithm may not have the time to explore all the hills before the individuals on the tallest hill takes over the population. This situation is called premature convergence, and is a primary cause of suboptimal solutions form genetic algorithms.

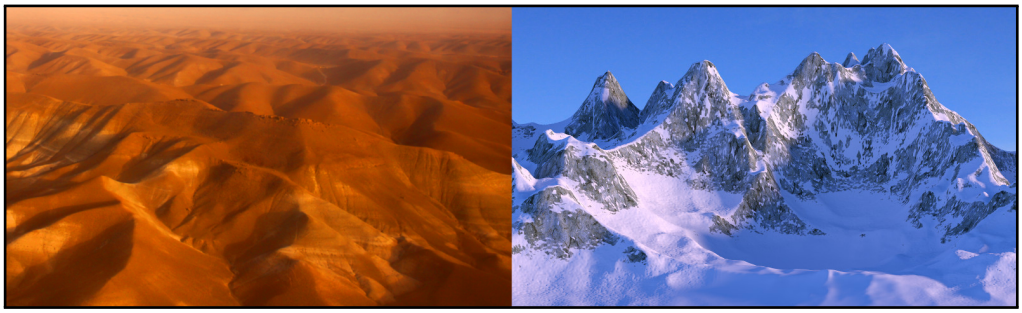


Fig. 11: Difference between solution spaces. Left, a landscape with several local maxima where it is difficult to find the tallest hill, and right, a landscape with local maxima but a clear global maximum.
Left: CC-BY-2.0 Ed Ledford, Right: CC-BY-2.0 Glenn.

The genetic algorithm excels if the search space is of such a structure that the combination of two locally optimal solutions in a crossover operation produces an even better offspring. Indeed, the traveling salesman problem has this property. Imagine two possible solutions to the same TSP, one is neatly arranged in the left half of the plane but a mess in the right half, the other is oppositely messed up in the left part and neat in the right. If a crossover operation were to cut the left half from the first solution and glue it together with the right half of the other a better solution would result.

3.1 Representations

The discussion of how hilly the search space is naturally depends on the optimization problem itself, but it also dependent on the genetic representation of the problem combined with the available crossover and mutation operators of the algorithm. We will have a look at some possible representations and operators for the TSP, and then discuss the form of the possible spaces.

The traditional genetic representation used by Holland is the binary representation, which is simply a binary string of zeros and ones. Holland also proposed two very easily understandable operators. The mutation operator will, for every bit in the genome, evaluate a stochastic process and if it is true, toggle the corresponding bit. His crossover

operator is given two parent genomes, it then chooses a random index in the string and the offspring is composed of the bits left of the index from one parent and the bits right of the index from the other. The mutation operator has the very nice feature that it may potentially transform any possible solution into the optimal in one shot by choosing all the wrong bits and flipping them. Thus the entire search space is fully connected and there is in fact no disjoint hills for the algorithm to get caught on. However, no obvious translation from the binary representation to TSP solutions is known. One naïve interpretation is to simply translate the sequences of waypoint indices to a binary form like

$$\{1\ 2\ 3\ 4\ 5\} \sim \{001\ 010\ 011\ 100\ 101\}\ .$$

However, neither of the two classical operators produce meaningful results. As put by Whitey et al. [24]

> Unfortunately, there is no practical way to encode a TSP as a binary string that does not have ordering dependencies or to which operators can be applied in a meaningful fashion. Simply crossing strings of cities produces duplicates and omissions. Thus, to solve this problem some variation on standard genetic crossover must be used. The ideal recombination operator should recombine critical information from the parent structures in a non-destructive, meaningful manner.

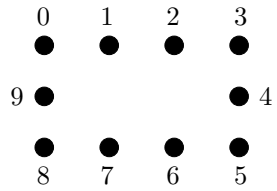
One of the most often used solutions to this problem is to not translate the list of indices into binary form but rather work on a list of integers. This is called the path representation. With this representation, the classical mutation operator is rendered unusable as there is no notion of flipping an integer. However, the obvious conversion to simply choose a random integer instead of flipping the bit yields the same problems as for the binary representation that it would produce infeasible solutions. The classical crossover does not produce feasible output either, but a good many variations of the mutation and crossover operators have been developed for the path representation, which makes this representation one of the most used today.

One mutation which is guaranteed to produce valid solutions is the inversion mutation, which Holland actually proposed himself albeit in the context of the binary representation. The mutation chooses two cut-points in the genome and inverts the sequence in between them. Intuitively, this is easy to understand in the path representation, as the direction of travel in the subsection is simply reversed while the rest of the solution is traversed as before. The inversion mutation is illustrated along with three other mutations in figure 12. The discussion about the hilly search space becomes relevant now, as the inversion mutation cannot, in one operation, move from any solution to the optimal. Instead, the mutation needs several steps to climb down the hill and cross the valley before ascending the optimal hill. Although the stochastic nature of the genetic algorithm permits this, the case becomes increasingly improbable the

further down the hill the solution has to move. Given enough repetitions however, the inversion mutation may mutate any arbitrary solution into the optimal. The proof for this is given in Paper C along with the same proof for the exchange and displacement mutation operators, which we will look at next.

The exchange mutation proposed by W. Banzhaf chooses two entries in the genome and exchanges them [25]. This mutation also guarantees feasible output and further it has the feature that all entries but the two affected are conserved in the same position in the genome. For the traditional TSP, however, this is not a very useful feature as the solution is a tour with no starting or ending point. Thus any path representation may be cyclically rotated without any change to the solution. On the other hand, this feature may be very convenient in scheduling problems. The exchange operator has the same property as the inversion operator that it may convert any solution into the optimal given enough iterations. But where the inversion had an intuitive physical representation the exchange is no so obvious; if one entry is out of place, why should we choose another (maybe perfectly placed) entry to exchange it with? This feature hinders the exchange mutation in perfecting almost optimal solutions, i.e. it is creating hills with many local optima.

Example 1 (Exchange Mutation Creating Local Maxima)
Given the TSP:



The solution {0 1 6 2 3 4 5 7 8 9} is very close to optimal, only the entry '6' is out of place. However, no exchange of two entries will reduce the length of the tour. A possible sequence of exchanges to achieve the optimal is:

exchange 4 and 6: {0 1 4 2 3 6 5 7 8 9}
exchange 2 and 4: {0 1 2 4 3 6 5 7 8 9}
exchange 5 and 6: {0 1 2 4 3 5 6 7 8 9}
exchange 3 and 4: {0 1 2 3 4 5 6 7 8 9}

This way, the solution steps one step down the hill and three steps up to the optimal solution. Other sequences exist, but all include the solution stepping down the hill.

The example shows how the exchange algorithm creates local optima around the very top of the hill.

A solution to this local optimum problem is to simply move only one entry. This method is known as the insertion mutation, proposed by D. Fogel [26]. In this mutation, the entry chosen is removed from its original position and inserted in any random position. The idea of the insertion mutation is generalized in the displacement mutation, described by Z. Michaelevicz [27], where a subsection of the genome is chosen and moved to a new location. The displacement mutation has the intuitive function that a section of the solution might be well-formed but located among otherwise unrelated entries can be moved to a more appropriate location in the genome.

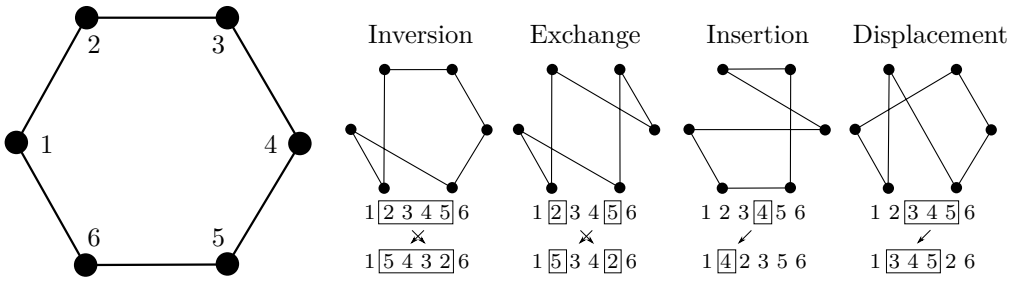


Fig. 12: Illustration of some mutation operations on a 6-city problem.

Even though the different mutations does not seem as superior as the classical mutation, that is able to potentially transform any solution into the optimal. However, such ability is not necessarily an advantage. Actually, the classical mutation may be regarded as a random search biased towards the parenting genome. The possible genomes resulting from a mutation is sometimes called the neighborhood of the genome. If the neighborhood of the genome is the entire search space, the evolutionary pressure of selection is somewhat reduced. The smaller the neighborhood, the better the algorithm is at hill-climbing but worse at exploring the different hills. The ability of hill-climbing, also called exploitation, is an attractive feature of the algorithm, but so is the exploration feature, so some trade off must be found when implementing the algorithm.

In Paper C the neighborhood size of the inversion, exchange, and displacement operators are evaluated using an entropy perspective. The entropy rate of a stochastic process is a measure of how much it transforms a sequence in terms to information.

Several crossover operators have also been developed specifically for the path representation. Also here, is it important to note how much change the operator brings into the offspring. It may however be a bit more difficult to measure. Conceptually, the least invasive operator should cut some information bearing from one parent and the rest from the other and not introduce any randomness. However, this may be impossible as the different parts of the parents must be glued together, which usually leads to using new edges between the nodes.

One of the crossover operators is the order crossover by Lawrence Davis [28], who first applied the mutation in a bin packing problem. The problem of bin packing is a matter of packing as many of a set of differently sized rectangles as possible into another rectangle. The bin packing problem and the TSP share the property of being epistatic domains, i.e. changes in one gene may affect how another gene is expressed. An example is the gene that codes for baldness in humans takes precedence over the gene that determines the color of the hair. In the bin packing problem, the rectangles packed first influences how and if the following rectangles can be packed, put another way, the order of the entries matter. The same is true for the path representation, the order of the entries is actually more important than the absolute position in the genome, which as noted before actually has no relevance. The order crossover starts by copying a subsection of one of the parent genomes to the offspring. The rest of the offspring is filled by copying the entries of the other parent while omitting the entries already in the copied subsection. To minimize new edges, the copying process from the second parent starts from the point which is equal to the last point in the subsection. Thus the resulting offspring maintains a high degree of the ordering in its parents.

Example 2 (Order Crossover)
The two parent genomes

$$\{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\} \text{ and } \{6\ 3\ 1\ 7\ 8\ 9\ 2\ 5\ 4\}$$

are combined, choosing the subsection $\{4\ 5\ 6\}$ from the first parent, which produces the offspring

$$\{8\ 9\ 2\ 4\ 5\ 6\ 3\ 1\ 7\}\ .$$

These operators may be implemented directly or modified to degrees suiting the programmer; sometimes a change in the structure of the operator causes a significant speed-up in the runtime of the algorithm. This way more evaluations can be executed in the same time, which may be more interesting than maintaining a specific structure.

Grefenstette et al. developed the ordinal representation in an attempt to allow the classical crossover operator [29]. The ordinal representation is constructed by subtracting the entries from a standard list while noting the indices from the resulting list in the genome.

Example 3 (Ordinal Representaion)
Given the standard list (1 2 3 4 5); the tour {4 1 3 5 2} would be constructed as

$$\begin{aligned} \{\} &\quad (1\ 2\ 3\ 4\ 5) \\ \{4\} &\quad (1\ 2\ 3\ 5) \\ \{4\ 1\} &\quad (2\ 3\ 5) \\ \{4\ 1\ 2\} &\quad (2\ 5) \\ \{4\ 1\ 2\ 2\} &\quad (2) \\ \{4\ 1\ 2\ 2\ 1\} &\quad () \ . \end{aligned}$$

However, the Grefenstette et al. themselves refuted the usefulness of the representation, as the path after the first cut point did not bear much resemblance with its parents.

A subset of all the different operators and representations have been described here. In [30], Larrañaga et al. provides a fine overview of several of the operators and representations for use in genetic algorithms solving traveling salesman problems.

3.2 Fitness Functions

The fitness function evaluates the genomes of the genetic algorithm to figure out how fit their phenotypes are with regard to the problem. The fitness is not necessarily the same as the cost of the objective function. Naturally, the fitness of a solution to a TSP should be higher for a lesser cost, so the fitness function could simply be multiplication with -1 or the reciprocal of the cost. These two approaches has a problem though. When the population converges to a solution, the costs of the individual solutions are rather much alike and the selection pressure is disabled leading to a random search. One thing to note about the behavior of the genetic algorithm is that the individuals are not evolving to achieve the global optimum but rather competing with the other individuals to dominate the population. This leads to a possible function that normalize the fitness according to the worst individual in the population such that the fitness of the i^{th} individual is

$$f_i = c_{\text{worst}} - c_i \ , \tag{1}$$

where c_{worst} is the objective cost of the worst individual in the population and c_i the cost of individual i.

The fitness function can also be used in an effort to control the composition of the population by using non-linear scaling. To increase exploration of the search space, the fitness function can be constructed so that the fitness of the lower and higher cost individuals are leveled out. This way, the worse individuals are selected more often, and

the population will be likely to diversify. Conversely, the best solutions may be given a higher weight to intensify the exploitation of that specific area of the search space.

3.3 Managing Populations

In order to avoid premature convergence, operators with a large neighborhood can be used to enhance exploration, which on the other hand limits exploitation. Another approach is to actively manage the population diversity. This approach follows the old advice of not putting all one's eggs in one basket. This way, the algorithm routinely checks the diversity of the population and if all the individuals are too alike they are probably exploring the same hill and the algorithm takes steps to diversify the population.

The question is how to measure the diversity of the population. Tsujimura and Gen proposed a method derived from entropy measures from information theory [31]. The entropy measure of a given information source is computed as a summation over all states i

$$H = \sum_i -p_i \log p_i \tag{2}$$

where p_i is the probability of the source emitting the state i. If the source only emits a single state, the entropy H is 0, and the higher the randomness the higher the entropy. Tsujimura and Gen proposed to count the appearances of the individual waypoints in the different positions of the genome, so that p_{wi} would represent the probability of waypoint w in location i yielding the entropy

$$H = \sum_{i \in I} \sum_{w \in W} -p_{wi} \log p_{wi} \tag{3}$$

where W is the set of all waypoints and I is the set of locations in the genome, and the individual probabilities are

$$p_{wi} = c_{wi}/N_p \tag{4}$$

where c_{wi} is the count of waypoint w in location i, and N_p is the size of the population.

Unfortunately this diversity measure is based on the absolute position of waypoints in the genome, but as noted earlier, the path representation is order based, so absolute positions are not ideal. Indeed, the two solutions

$$\{1\ 2\ 3\ 4\} \text{ and } \{3\ 4\ 1\ 2\}$$

are the same and should produce a 100 % likeness measure. One way to achieve this, is to evaluate the adjacency matrix of the solution rather than the genome itself. The

solutions above produces the same adjacency matrix

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} .$$

By adding all the adjacency matrices of the population and multiplying the result by the reciprocal of the population size, the entropy diversity of the population can be computed.

Example 4 (Entropy of Adjacency Matrix)
The two above mentioned solutions should produce an entropy of 0. This is checked by producing the probability adjacency matrix

$$M = \frac{1}{2}\left(\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{pmatrix} .$$

We get the entropy by iterating over the indices $m_{kl}$of the matrix M

$$H = \sum_k \sum_l -m_{kl} \log m_{kl} = 0 .$$

A similar approach can be applied to the mutation operators to get a measure for the size of its neighborhood. Such an analysis is carried out in Paper C. When a stochastic process modifies an information sequence, the amount of random information that is added to the sequence is called the entropy rate. By applying the above outlined calculation to the input and output genome given to and received from a mutation, its entropy rate is obtained. By this measure, if a mutation has a large neighborhood it also has a high entropy rate. This knowledge can be used to diversify a population of high similarity by applying mutations with high entropy rate to increase the exploration, conversely, it may apply less disruptive mutations in the final phases of the run to exploit the found hills. The results from Paper C is summarized in figure 13.

Elitism

Elitism is somewhat the opposite of diversifying the population. It is a method to constrain the exploration of the algorithm. It works by transferring the best individuals from a parenting generation to the following generation. This ensures that the best found solution is exploited. Even though the selection of the fittest individuals probabilistically guarantees that the best solution is exploited, the mutated best individual may not move

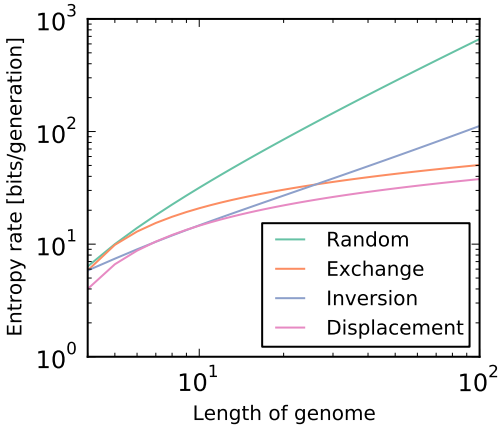


Fig. 13: Entropy rates of the three mutations described, compared to the totally random mutation. Note that these figures are for a directed TSP.

up the hill. In the case this happens and no improvement happens to the best solution, the elitism approach inserts the best known individual into the new generation.

There are several approaches to elitism, one is to transfer the best individual if no new individual is more fit. Another approach proposed by Deb et al. in their NSEA-II algorithm is to keep a record of the M best known individuals, and let these participate in the selection process, so that the basis for the selection is among $N + M$ individuals where N is the population size [32]. An also used variant, which combines the two, is to replace the worst M individuals in the new generation with the archived elite.

3.4 Terminating the Algorithm

If nothing stops the genetic algorithm, there is no reason why it cannot keep generating new generations forever. The usual way of stopping the algorithm is to evaluate the population against a termination criterion and terminate the computation if the criterion evaluates to true.

The basic termination criterion is the *max generations* criterion, which simply sets a maximum number of generations to be evaluated. This criterion is usually incorporated as a failsafe to ensure that the algorithm does stop within some time frame.

More intelligent are the the two traditional stall criteria *stall-generations* and *stall-time*. As the name implies, these criteria monitors if the algorithm is stalling, that is if the improvement of the individuals are stagnating. If the stall-generations does not see an improvement in the fitness of more than a given threshold within a given number of generations, it will stop the algorithm. the same is true for the time-stall criterion, but instead of a number of generations, the criterion evaluates over a given time frame.

These termination criteria are easily understood, however some insight into the genetic algorithm is needed to set sensible values. Specifically, it is important to know about how much improvement can be expected per generation, otherwise the generation or time frame may be set too short so that the algorithm terminates while it is actually making viable progress. Conversely, if the frame is set too long, computational time is wasted.

In Paper D an adaptive parameter free termination criterion is presented. This criterion takes advantage of the fact that the genetic algorithm is solving a TSP with the objective of bringing down the execution time of the resulting solution. It basically stops the algorithm is it predicts that it will take more time to find a better solution than to actually fly the sub-optimal solution.

The prediction is based on the observation that the fitness of the best individual in solving a TSP usually follows the same pattern. This pattern was observed by D. B. Fogel to be "generally logarithmic" [26]. However, the curve seems to fit better with a polynomial, which lead to development of the model described in Paper D.

3.5 Infeasible Solutions

When the problems that the genetic algorithm solves involves constraints, the mutations sometimes produce infeasible solutions. This was the case in the afore mentioned case of binary representation and classical operators.

Example 5 (Producing infeasible solutions.)
The two solutions

$$\{1\ 2\ 3\ 4\ 5\} \sim \{001\ 010\ 011\ 100\ 101\}$$
$$\{2\ 4\ 5\ 1\ 3\} \sim \{010\ 100\ 101\ 001\ 011\}$$

crossed so that bits [3, 8] is taken from the first solution and the rest from the second produces.

$$\{011\ 010\ 011\ 001\ 011\} \sim \{3\ 2\ 3\ 1\ 3\}$$

which is not a feasible solution.

One approach to remedy these malformed genomes is to perform a reparation procedure on all infeasible offspring, which ensures that all individuals keep within the bounds of the constraints. A simple reparation for the problem in example 5 is to traverse the genome, remove any duplicates, and append any missing nodes at the end. Care must be taken that the reparation does not introduce too much randomization, mutating the genome and spoiling the likeness between the parents and the offspring, thus working

against the purpose of the crossover operator. Indeed, the reparation function just outlined is a very poor candidate.

The character of the infeasibility above stems from the inappropriate representation. Other constraint violations are more fittingly handled by reparation procedures. An example may be if the algorithm is optimizing the speed profile of a helicopter in a turn. Then any mutation to the target speed which exceeds the maximum speed of the helicopter may simply be capped at that maximum speed. Such a reparation conserves quite some likeness between the infeasible and the repaired solution.

In [33], Ray et al. note that

> Real life optimization problems often involve one or more constraints and in most cases, the optimal solutions to such problems lie on constraint boundaries.

Indeed, this is the case in linear programming. The simplex algorithm is often used to solve linear programs, and it works by evaluating the vertices that is generated by the intersections of the hyperplanes that the constraints produce in the solution space. This does not necessarily apply for non-linear problems. However, it may seem intuitive that in the case of speed profiling a turn, the fastest turn lies on the boundary defined by the maximum speed of the helicopter. This spawned the development of the infeasibility driven evolutionary algorithm (IDEA) of Ray et al. [33], which allows some degree of infeasibility, so that the neighborhood of the optimal solution extends into the infeasible region as well as into the feasible region. This way, there are more possible trajectories for the solutions to enter into region around the optimum.

The use of infeasible genomes have been explored in Paper E, where the fuel usage of the helicopter is constrained to its fuel capacity. The job of the genetic algorithm is to optimize the sequence of waypoints as usual, but also to insert refueling stops at the appropriate times, so that the helicopter does not run dry. The approach here is to allow some over-fueling in such a way that the distances between the refueling stops may break the constraint. The idea is to exploit the symmetry that exists when inserting a refueling stop into a sequence: If the sequence is feasible, some fuel may be left in the tank and a displacement mutation may be able to put in another waypoint into the sequence. Conversely, if the sequence is infeasible, the displacement mutation simply needs to remove the waypoint. This way, the algorithm is more free to move around in the search space, and may converge to the optimum faster.

3.6 Multi-Variable and Multi-Objective Optimization

The concepts of multi-variable and multi-objective optimization sounds alike but represents two rather different topics. Multi-variable optimization is when several variables are tuned to optimize the objective. This is what happens in Paper F, where we are both optimizing the sequence of the waypoints, the heading of each waypoint, and their

individual speed. This is done to construct a trajectory composed of Dubins curves which are augmented to incorporate different turning radii as a function of the target speed of the waypoints. As more variables are added, the dimensionality of the search space increases, usually leading to longer search times.

Multi-objective optimization is a matter of optimizing several objectives no matter the number of variables. The solution to such problems is not necessarily a single solution as for the single-objective case, but rather a set of Pareto optimal solutions. In the case of the problem in Paper F, the optimization objective was to minimize time usage. When the target speeds of the waypoints are increased, the total runtime tends to decrease, but because of the following larger turning radii, the total path length increases. If the problem was made to be multi-objective by including the path length as a parameter as well, the solution would be the Pareto front defined on the time versus length plane.

The NSEA-II algorithm by Deb et al. [32] is developed to solve multi-objective optimization problems. The complexity of this algorithm is $\mathcal{O}\left(MN^2\right)$ where M is the number of objectives and N the population size. The reason for this is that it has to evaluate all new individuals in the new population against the set of Pareto optimal solutions. (In the section on elitism, it was mentioned that NSEA-II includes all the best known solutions in the selection process, these best known solutions are in fact the candidate Pareto optimal solutions.)

3.7 Using Genetic Algorithms for TSP

But why at all use the genetic algorithm for solving the TSP when the Dantzig-Fulkerson-Johnson and Lin-Kernighan algorithms solve problems with thousands of cities? Well, several of the reasons have been outlined. The genetic algorithm has features that is not possible to incorporate in the other algorithms.

The genetic algorithm is an anytime algorithm; at any time it can present a feasible solution. This enables the ASETA system to receive a set of waypoints for the unmanned aircraft to visit, quickly perform an initial planning and dispatch the aircraft to the first waypoint in the plan, and while the aircraft is flying perfect the plan for when it needs the next point.

When the aircraft collects information and automatically generates new waypoints, they are easily appended to the problem formulation, and mutation operators like the displacement operator can move the waypoint into a good position in an otherwise quite good sequence.

The genetic algorithm is not limited to only optimize the sequence of the TSP. It can incorporate more advanced models of the aircraft through the objective function and also optimize speeds and headings of the waypoints.

The genetic algorithm can plan in problems with limited resources like fuel capacity. This leads to solutions of varying lengths with several possible refueling stops; an area

that genetic algorithms actually handles very gracefully as the genomes simply has different lengths, which is handled by most operators without much modification.

Being population based, the genetic algorithm provides a possibility to sample the Paretto front of a multi-objective optimization problem, so that several factors may be optimized simultaneously, such as time and fuel consumption.

Basically, the genetic algorithm provides a very extensible basis for building a planner that takes many factors and models into account and is quite robust to changing conditions.

References

[1] D. Slaughter, D. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, no. 1, pp. 63–78, Apr. 2008.

[2] J. Plovsing and K. Hjulsager, "Landbrug 2011 - Statistik om landbrug, gartneri og skovbrug," Danmarks Statistik, Tech. Rep., 2011.

[3] A. Jones, P. Panagos, S. Barcelo, F. Bouraoui, C. Bosco, O. Dewitte, C. Gardi, M. Erhard, J. Hervás, R. Hiederer, S. Jeffery, A. Lükewille, L. Marmo, L. Montanarella, C. Olazábal, J.-E. Petersen, V. Penizek, T. Strassburger, G. Tóth, M. V. D. Eeckhaut, M. V. Liedekerke, F. Verheijen, E. Viestova, and Y. Yigini, "The State of Soil in Europe," European Commission Joint Research Centre, Luxembourg, Tech. Rep., 2012.

[4] S. K. Mathiassen, T. Bak, S. Christensen, and P. Kudsk, "The Effect of Laser Treatment as a Weed Control Method," *Biosystems Engineering*, vol. 95, no. 4, pp. 497–505, Dec. 2006.

[5] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[6] R. C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm," Carnegie Mellon University, Tech. Rep., Jan. 1992.

[7] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, no. 3, p. 497, Jul. 1957.

[8] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, Mar. 2001.

[9] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins' vehicle," in *Proceedings of the 2005, American Control Conference, 2005.* Portland, OR, USA: IEEE, 2005, pp. 786–791.

[10] A. Tsourdos, B. White, and M. Shanmugavel, *Wiley: Cooperative Path Planning of Unmanned Aerial Vehicles.* John Wiley & Sons, Ltd, 2011.

[11] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum Press, 1972, pp. 85–103.

[12] K. Menger, "Bericht über ein mathematisches Kolloquium 1929/30," *Monatshefte für Mathematik und Physik*, vol. 38, no. 1, pp. 17–38, Dec. 1931.

[13] G. B. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a Large-Scale Traveling-Salesman Problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[14] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems," *Mathematical Programming*, vol. 97, no. 1, pp. 91–153, 2003.

[15] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[16] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[18] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[19] M. Gendreau and J.-Y. Potvin, Eds., *Handbook of Metaheuristics*, 2nd ed., ser. International Series in Operations Research & Management Science. Boston, MA: Springer New York Dordrecht Heidelberg London, 2010, vol. 146.

[20] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[21] S. Thrun, "Probabilistic robotics," 2002.

[22] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, MA, USA: MIT Press, 2005.

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[24] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and traveling salesmen: The genetic edge recombination operator," *Proceedings on the Third International Conference on Genetic Algorithms*, pp. 133–140, 1989.

[25] W. Banzhaf, "The "molecular" traveling salesman," *Biological Cybernetics*, vol. 64, no. 1, pp. 7–14, Nov. 1990.

[26] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, vol. 60, no. 2, pp. 139–144, Dec. 1988.

[27] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Springer-Verlag, 1996.

[28] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proceedings of the 9th International Joint Conference on Artificial Intelligence, IJCAI'85*, vol. 1. Morgan Kaufmann Publishers Inc., Aug. 1985, pp. 162–164.

[29] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. V. Gucht, "Genetic Algorithms for the Traveling Salesman Problem," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Pittsburg, 1985, pp. 160–165.

[30] P. Larrañaga, C. M. Kuijpers, R. H. Murga, I. n. Inza, and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[31] Y. Tsujimura and M. Gen, "Entropy-based genetic algorithm for solving TSP," in *Proceedings of the Second International Conference on Knowledge-Based Intelligent Electronic Systems*, vol. 2, no. April. IEEE, 1998, pp. 285–290.

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[33] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, "Infeasibility Driven Evolutionary Algorithm for Constrained Optimization," in *Constraint-Handling in Evolutionary Optimization*, E. Mezura-Montes, Ed. Springer Berlin Heidelberg, 2009, pp. 145–165.

Part II

Papers

Paper A

Adaptive Surveying and Early Treatment of Crops with a Team of Autonomous Vehicles

Wajahat Kazmi, Morten Bisgaard, Francisco Garcia-Ruiz,
Karl Damkjær Hansen, Anders la Cour-Harbo

The paper is published in the
Proceedings of the 5th European Conference on Mobile Robots, ECMR 2011

Abstract

The ASETA project (acronym for Adaptive Surveying and Early treatment of crops with a Team of Autonomous vehicles) is a multi-disciplinary project combining cooperating airborne and ground-based vehicles with advanced sensors and automated analysis to implement a smart treatment of weeds in agricultural fields. The purpose is to control and reduce the amount of herbicides, consumed energy and vehicle emissions in the weed detection and treatment process, thus reducing the environmental impact. The project addresses this issue through a closed loop cooperation among a team of unmanned aircraft system (UAS) and unmanned ground vehicles (UGV) with advanced vision sensors for 3D and multispectral imaging. This paper presents the scientific and technological challenges in the project, which include multivehicle estimation and guidance, heterogeneous multi-agent systems, task generation and allocation, remote sensing and 3D computer vision.

1 Introduction

Weeds have always remained a major concern to farmers because they compete with crops for sunlight, water and nutrients. If not controlled, they can cause a potential loss to the monetary production value exceeding a global average of 34% [1].

Classical methods for weed removal are manual or mechanical which are time consuming and expensive. Over the last few decades, herbicide application has been a dominant practice. Indiscriminate use of chemicals, on the other hand, is also detrimental to both environment and the crop itself.

Reduction in the use of pesticides in farming to an economically and ecologically acceptable level is one of the major challenges of not just developed countries but also the developing countries of the world. Introducing an upper threshold to the amount of pesticides used does not necessarily serve the purpose. It must be accompanied with the knowledge of when and where to apply them. This is known as Site-Specific Weed Management (SSWM). For SSWM, the concept of precision farming scales down to field spots or patches [2] or even to plant scale [3]. This requires real-time intelligence on crop parameters which significantly increases the complexity of modern production systems and therefore imply the use of automation through information technologies, smart sensors and decision support systems.

Over the last five decades, the concept of agricultural automation has evolved from mechanization of manual labor into intelligent sensor based fully autonomous precision farming systems. It started with automation of ground vehicles [4] and over time, air vehicles also found their way in. Furthermore, advanced perception technologies such as machine vision have become an important part of agricultural automation and 2D/3D image analysis and multispectral imaging have been very well researched in agriculture.

Today, with advanced sensor technologies and both air and ground, manned and unmanned vehicles available in the market, each one with its own pros and cons, the choice has become broad. The technology is at par with most of the industrial demands but the need is of an optimal subset of technical attributes since the practice, particularly in agriculture, has usually been limited to the use of one type of vehicle with a limited sensor suite. The drawback of this scheme is that one type of vehicle is unable to satisfy all operational requirements. For example an unmanned aircraft (UA) to detect and apply spray to the aquatic weeds compromises on spray volume, precision and duration of flight due to weight-size constraints [5], while a ground vehicle alone can significantly slow down the operation along with producing substantial soil impact [6], not to mention the problem of emissions.

These constraints imply the use of a team of both air and ground vehicles for a holistic solution. Unmanned (ground) vehicles being considerably smaller in size than manned vehicles have lesser soil impact and fuel consumption (thus have reduced emissions) and may also be battery operated. Therefore, for economy of time and energy and for higher precision, a network of unmanned air and ground vehicles is inevitable and is destined to outperform conventional systems. Research has also been conducted in cooperative unmanned mixed robotic systems both for civil and military purposes, for example, [7] proposes hierarchial framework for a mixed team of UAS and UGV for wildfire fighting and GRASP laboratory [8] used such systems in urban environments as a part of MARS2020 project. But apparently, no such strategy has been adopted in agriculture. To the best of authors' knowledge, ASETA is the first project of its kind to use a team of both UAS and UGV in agriculture which has opened a new chapter in precision farming and researchers especially in the European Union are taking increased interest in such approaches (for example, RHEA project [9]).

This paper describes the scope of ASETA's scientific research, its heterogeneous robotic fleet and sensor suite for SSWM. The paper is organized as follows: the project is described in section 2, followed by equipment summary in section 3. Main research areas of this project in the context of the related work are presented in section 4. Section 5 concludes the paper.

2 ASETA

ASETA (Adaptive Surveying and Early treatment of crops with a Team of Autonomous vehicles) is funded through a grant of 2 million EUR by the Danish Council of Strategic Research. It aims at developing new methods for automating the process of acquiring and using information about weed infestation for an early and targeted treatment. The project is based on the following four hypotheses:

1. Localized detection and treatment for weeds will significantly decrease the need for herbicides and fuel and thereby reduce environmental impact.

2. Such early detection can be accomplished by multi-scale sensing of the crop fields by having UAS surveying the field and then performing closer inspection of detected anomalies.

3. A team of UAS and UGV can be guided to make close-to-crop measurements and to apply targeted treatment on infested areas.

4. A team of relatively few vehicles can be made to perform high level tasks through close cooperation and thereby achieve what no one vehicle can accomplish alone.

Fig. A.1: ASETA Strategy

The strategy adopted in ASETA (Fig. A.1) is to survey crop fields using UAS in order to obtain and localize hotspots (infested areas) through multispectral imaging followed by cooperative team action among a team of air and ground vehicles for a closer 3D visual inspection, leading to the treatment. Survey may be iterated depending on the team size and field dimensions.

Obviously, ASETA's industrial gains come at the cost of certain technical and scientific challenges. A heterogeneous team of several unmanned vehicles is chosen to distribute heavy payloads on ground vehicles (sensing, perception and treatment) and relatively lighter payload (sensing and perception only) on the air vehicles which potentially is a well balanced approach but puts high demands on team cooperation and task management keeping in view the constraints of each team member. A further complexity to the proposed system arises from the fact that although computer vision is very popular and successful in plant inspection, however, changing weather and sunlight conditions has so far limited in-field agricultural vision systems [10]. These challenges must be addressed in order to produce an optimal combination of more than one type of unmanned vehicles to outperform the conventional systems in the scope. Therefore, in order to achieve its goals, ASETA will carry forward scientific research in four directions,

namely, multispectral imaging, 3D computer vision, task management and multivehicle cooperation.

The project started in January 2010. Major research work will be carried out from 2011 to 2013. Scientific research is being conducted by four post graduates and several faculty staff involved at two Danish universities, University of Copenhagen and Aalborg University. This collaborative work is a mixture of theory, simulations, and actual fields tests. The latter is done in cooperation with the university farms at University of Copenhagen, which will maintain a field of sugar beets throughout the growth seasons in 2011 to 2014. Since sugar beet is the crop-of-choice for the demonstrative part, Nordic Beet Research is also involved in the project.

3 Equipment

Some of the specialized equipment used in this project is described below:

3.1 Robotic Platforms

ASETA has three unmanned mobile robots available for the project. They are briefly described below:

UAS

The UAS is comprised of two rotary wing aircraft. The first UA is a modified Vario XLC helicopter with a JetCat SPTH-5 turbine engine (Fig. A.2). The helicopter weighs 26 kg when fully equipped for autonomous flight and can fly for 30 minutes with 6 kg of fuel and 7 kg of payload. For autonomous flight, a NAV440 Inertial Navigation System (INS) from Crossbow is used together with altitude sonar. Onboard computer is a Mini-ITX with dual-core 1.6 GHz Intel Atom processor and runs a Debian Linux operating system. The flight time in this configuration is approximately 30 minutes.

Fig. A.2: Autonomous vehicles in ASETA, (from left): Vario XLC, Maxi Joker-3 and robuROC-4

The second UA is a modified Maxi Joker-3 helicopter from MiniCopter. It is electrically powered and weighs 11 kg when equipped for autonomous flight (Fig. A.2). The

helicopter can fly for 15 minutes with a payload of 3 kg. It has a Xsens MTiG INS and sonar altimeters for autonomous flight and Nano-ITX size 1.3 GHz onboard computer with Debian Linux operating system.

Each UA can be configured to carry the multispectral camera (see Section 3.2) or a color camera. The sensors are mounted in a modified OTUS L205 gimbal from DST Control. The low level guidance, navigation, and control (GNC) system for the UAS is the baseline GNC software from Aalborg University's UAV lab[1]. It features gain scheduled optimal controller, unscented Kalman filter for navigation and an advanced trajectory generator.

UGV

The ground vehicle is a robuROC-4 from Robosoft (Fig. A.2). Running on electric power this vehicle is designed for in-field use and will carry the TOF (see Section 3.2) and color cameras for close-to-crop inspection. The total weight is 140 kg (without vision system) and it is controlled by a standard laptop residing under the top lid running the cross-platform robot device interface Player/Stage. This vehicle is equipped with RTK GPS to allow it to traverse the crop rows with sufficient accuracy.

3.2 Vision Systems

As described in section 2, two different imaging systems will be used: one for remote sensing and another for the ground based close-to-crop imaging. For remote sensing, a multispectral camera will be employed and for ground based imaging a fusion of Time-of-Flight and color images will be explored.

Multispectral Camera

The multispectral camera used in the project is a Mini MCA from Tetracam[2] (Fig. A.3). This specific sensor weighs 695 g and consists of six digital cameras arranged in an array. Each of the cameras is equipped with a 1.3 megapixel CMOS sensor with individual band pass filters. The spectrometer filters used in this project are 488, 550, 610, 675, 780 and 940 nm (bandwidths of 10 nm). The camera is controlled from the on-board computer through an RS232 connection and images are retrieved through a USB interface. Video output is also possible using the output video signal in the control connector.

Time-of-Flight Camera

A time-of-flight (TOF) camera system has the advantage that depth information in a complete scene is captured with a single shot, thus taking care of correspondence

[1] www.uavlab.org
[2] www.tetracam.com

Fig. A.3: Mini MCA multispectral camera.

Fig. A.4: SwissRanger SR4000 TOF Camera

problem of stereo matching. In this project, Mesa Imaging's $SwissRanger^{TM}SR4000$ [3] USB camera will be used which is an industrial grade TOF camera allowing high quality measurements in demanding environments. It operates in the Near-InfraRed (NIR) band (illumination wavelength 850 nm) hence a stable measurement accuracy and repeatability can be achieved even under variations in object reflectivity and color characteristics. SR4000 can deliver a maximum frame rate of 50 frames/sec. As usually is the case with TOF cameras, the resolution is fairly low (176 x 144 pixels) which will be augmented by fusion with high resolution color images.

4 Research Areas

The main scientific contributions will be generated by four research positions associated with the ASETA loop (Fig. A.1). Two PhD studies in analysis and interpretation of images detection and treatment of weeds and one PhD study and one Post Doc in task allocation and vehicle cooperation. They are briefly described below in the context of the state-of-the-art.

[3] www.mesa-imaging.ch

4.1 Multispectral Aerial Imaging for Weed Detection

As already discussed in section 1, SSWM involves spraying weed patches according to weed species and densities in order to minimize herbicide use. However, a common approach in SSWM is weed mapping in crops which is still one of the major challenges. Remote sensing supplemented by targeted ground-based measurements have been widely used for mapping soil and crop conditions [11, 12]. Multispectral imaging at low and high spatial resolution (such as satellite and airborne) provide data for field survey and weed patch allocation but depending on the system used, it varies in accuracy [13].

A higher level of spectral difference between plant and soil makes their separation relatively easy in a multispectral image. But the spectral ambiguity among plant species makes plant classification a difficult task. Thus, the spatial resolution of the sensor becomes an essential criterion for a reliable vegetation discrimination in order to detect the spectral reflectance in least altered form to avoid spectral mixing at pixel level [14]. Therefore, the major requirements for robust aerial remote sensing for weed identification are a high spectral resolution with narrow spectral bands and the highest possible spatial resolution (normally limited by sensor technology) [15].

The high usability of multispectral satellite imagery from *QuickBird* (2.4 to 2.8 meter spatial resolution) in a sugar beet field for *Cirsium arvense* L. hotspot detection for a site-specific weed control having spot diameters higher than 0.7 m was demonstrated by [16]. The relatively low spatial resolution along with the inability to image ground during cloudy conditions make such systems less suitable for analyzing in-field spatial variability. On the other hand, high resolution images (up to 0.707 mm/pixel) were acquired in a rice crop for yield estimation using a UA flying at 20 m [12].

Keeping this fact in view, in this project, the choice of camera equipped unmanned helicopters is made because they can be guided at lower altitudes above the crop canopy in contrast to the satellite and manned airborne systems, increasing image resolution and reducing atmospheric effects on thermal images [17, 18]. Images obtained from low altitudes will support accurate decision making for precision weed and pest management of arable, tree and row crops.

The goal of aerial imaging in ASETA is to explore the potential of multispectral imaging involving multistage sampling for target detection meanwhile employing spatial sampling techniques (stereology) for real-time density estimation. Stereology will be used for target sampling at various scales, using information from lower resolution images (high altitude-helicopter) to plant measurements at higher resolutions (low altitude-helicopter) to maximize information from sparse samples in real-time while obeying rules of probability sampling [19]. The maps of the field provide the basis for optimal designs of sampling locations over several spatial scales using variance reduction techniques [19].

4.2 3D Computer Vision for Weed Detection

Multispectral aerial imaging will be able to detect hotspot locations and volumes, but on a macro level. It cannot resolve individual plants at intra-row level. A ground based imaging system will thus be employed for close-to-crop inspection in this project.

In agricultural automation, the expected outputs of a weed detection system are weed plant detection, classification and stem center localization. Ground based imaging is not new but research has mainly focused on weeds at very early growth stages. There are two main reasons for this; an early detection will lead to an early treatment and the fact that plant imaging and recognition is one of the most demanding tests of computer vision due to complicated plant structures and the occlusion of crop and weed plants at later stages of growth prevents the proper visual separation of individual plants. While some efforts have shown promise under conditioned environments such as green houses, lack of robust resolution of occlusions remains a major challenge for in-field systems [20]. By utilizing 3D visual information it becomes possible to detect occlusions and make a better visual separation. Keeping this fact in view, the major objective in this project in ground based imaging is to utilize 3D computer vision techniques in weed detection.

There has been a significant amount of research work done towards 3D analysis of plants as well, but again this has mainly been aimed at navigation in the field, in estimating overall canopy properties through stereovision or creating very detailed models of plants [10]. 3D modeling is computationally expensive and is potentially hampered by thin structures, surface discontinuities and lack of distinct object points such as corners ending up in the correspondence problem [21]. These limitations pose a major challenge for in-field real-time 3D analysis of plants.

In order to address these problems, active sensing system based on Time-of-Flight (TOF) technology will be used which has been very scantily tested in agricultural applications mainly due to a very high sensor cost. TOF has a drawback of low resolution and sensitivity to ambient light, but these problems have been recently addressed and having TOF depth map fused with high resolution color image has shown very encouraging results especially with parallelized computations which significantly reduces the runtime [22]. The idea, therefore, is to use TOF data integrated with high resolution color images to perform in-field plant analysis. TOF technology has only recently found its way towards industrial applications and in agricultural automation its utility assessment is quite fresh [23–25].

While 3D analysis is required for resolving occlusions and localization of plant body, discrimination of weeds from crops is still another challenge. Pattern and Object Recognition techniques have been widely used in weed discrimination [26]. But most of the techniques use color or size of the leaves (Leaf Area Index-LAI) as prime feature. The size of the leaves or the exposed area of the leaves vary due to orientation, growth stage and weather conditions. Furthermore, variations in the soil conditions and the amount of sunlight can result in color variations. Instead, vision systems based on shape are less

sensitive to variation in target object color [10]. In this project, a shape based approach in distinguishing sugar beet crop plants from weeds will be used, for example [27].

In general, ASETA will contribute a new approach in weed identification by combining TOF technology with pattern recognition techniques bringing the lab research to the field.

4.3 Task Management

The idea of Future Farms is that the farm manager should be able to–more or less–just press a button, and then leave it until the process is finished. This demands that the system is capable of identifying the subtasks contained in this high-level command and ensure their execution. These two processes are commonly known as *Task Decomposition* and *Task Allocation*.

The task decomposition process is going to break down the overall task to small manageable chunks, that the individual members (robots) of the system are able to execute. The decomposition depends on the combined set of capabilities of the members. For example, if a member has the capability to take very high resolution images, the initial images might be taken from high altitude and only a few overview images may be sufficient for mapping the the entire field. Whereas, if only low resolution cameras are available, several overview images may be required.

When the overall task has been decomposed into suitable subtasks, they must be distributed to each of the members in the system. This is known as Task Allocation. Several different approaches to this have been investigated. Two broad categories can be identified as centralized and distributed allocation. The centralized approach is essentially a matter of solving a multiple travelling salesman problem (m-TSP). The distributed approach will divide the task of solving the TSP between each member. In this case the members must communicate with each other to make sure that two members are not planning to visit the same point (see section 4.4).

The TSP solution has historically received a great deal of attention and has shown to be $\mathcal{NP}$-hard [28], thus simple brute-force algorithms will not be practically usable in the system. The Lin-Kernighan heuristic [29] of 1971 is still one of the most preferred algorithms for solving TSPs, and maintains the world record of solving the largest TSP [30]. A strategy to solve the TSP with timing constraints (TCTSP) is devised in [31]. Helicopters conducting a closer examination of the weed infestations in the ASETA scheme will experience a TCTSP as the high altitude images will be taken over time and thus the close-up tasks are time constrained. Walshaw proposed a multi-level approach for solving the TSP [32]. This is relevant as the high altitude-helicopter process coarsens the TSP for the low altitude-helicopter, and thus gives a coarse representation of the low-level TSP free of charge.

The decentralized approach relies on the members to distribute the tasks among themselves, without intervention of a governing system. The MURDOCH allocation

system uses an auctioning approach where each robot bids on the different tasks depending on their own perceived fitness for the task at hand [33]. The fitness assessment of the ALLIANCE architecture [34] is based on a impatience behavioral pattern. These approaches will not guarantee the optimal solution, but provide some robustness that might be missing in the centralized approach.

The aim of the ASETA task management is to utilize existing TSP solving methods such as Lin-Kernighan or Walshaw approach and adapt them to the situation at hand, with the members gradually revealing more and more information as they move closer to the crops, from the high altitude- over to the low altitude-helicopter down to the ground vehicle.

4.4 Multivehicle Cooperation

The close cooperation among team members (robots) is an important part of ASETA in order to ensure a safe and efficient execution of the tasks provided by the Task Management. The cooperation layer will determine which robot will tackle which task and to some extent in what order. In a situation where a team of heterogeneous robots must cooperate in order to complete a task in an open-ended environment, it is crucial that each member has a clear understanding of its own as well as the other members' capabilities because they are not equally qualified to handle a given task. In this project, The helicopters are equipped with several different types of sensors including cameras (as described in section 3) well suited for observation only and the ground vehicle has an altogether different sensor suite and is meant for closer inspection and treatment. This information is to be used by every member to decide which part of the overall task it should handle and how to do it.

To ensure a timely and efficient execution of the tasks it is equally important for a robot to know what its team members are doing – i.e. their behavior – and thereby ensuring that two members do not unnecessarily work on the same subtask. However, it is not always trivial to acquire such knowledge. The distances involved in field operations can potentially become very large and thus can only allow limited communication. Furthermore, when reducing necessary communication among members, backwards compatibility is made easier and this is preferable in a industrial product. Therefore, the members must be able to deduce this knowledge from very limited information such as the state (position, orientation, and velocity) of the other members. This will put lesser constraints on the robots that are allowed to participate in the cooperation. In fact even robots without any cooperative capabilities can be a part of the system, as long as they can share their state with the rest of the team.

Current research in cooperative control of multivehicle systems focuses mainly on the control element such as formation control or distributed optimization. A comprehensive review of recent research in cooperative control can be found in [35]. Only few projects

have taken the limited communication between robots into account (for example: [36] or [37]).

In this project, the actual cooperation layer is created as a decentralized two-level approach:

Level 1: Acquiring team behavioral information

The challenges of this level are seen primarily as a model based estimation problem which will be solved using particle filtering. This is done through the formulation of a behavioral modeling framework which in turn describes the different possible behaviors of the members. When used in a particle filter, it is capable of determining the maximum likelihood hypothesis, i.e. best fitting behavior of the observed team members.

Level 2: Task execution

Each member is assumed to be containing a low level navigation and control system as well as simple trajectory planning. As a high level control, a receding horizon is used in the form of a decentralized Model Predictive Controller (MPC). The MPC on each member will attempt to find an optimal behavioral action to take, given information about the current behavior of the rest of the team.

In short, the ASETA cooperation scheme will use particle filtering and model predictive control to implement cooperation between loosely coupled robots.

5 Conclusion

ASETA will not only produce high quality research in multispectral imaging, computer vision and multivehicle systems, but it also aims at developing an actual demonstrator. Working within the price range of other farming machinery and the use of off-the-shelf hardware throughout enhances the likelihood of tools developed in this project being adopted by the industry. The long term objective of ASETA is a commercially available autonomous multi-scale surveying system for site specific weed management to reduce the cost and environmental impact of farming chemicals, fuel consumption and emissions. It therefore holds the potential for significant impact on the future of precision farming worldwide.

Given the rising levels of atmospheric CO_2 and temperatures under climate change, weed species are expected to show a higher growth pattern than crops due to their greater genetic diversity [38]. On the other hand, governments mandate considerable reductions on the use of pesticides. This fact has added more importance and promise to such projects.

Although dealing with a system of heterogeneous vehicles increases the complexity of the system, however, it also serves as a flexibility on the user end in the choice of

vehicles and sensors from a wide range, producing a more customized solution to the application at hand. ASETA, therefore, has future beyond agriculture towards several other applications such as fire fighting, search & rescue and geological surveying, in the long run.

References

[1] E. Oerke, "Crop losses to pests," *The Journal of Agricultural Science*, vol. 144, no. 01, pp. 31–43, 2006.

[2] S. Christensen, T. Heisel, A. M. Walter, and E. Graglia, "A decision algorithm for patch spraying," *Weed Research*, vol. 43, no. 4, pp. 276–284, 2003.

[3] M. Ehsani, S. Upadhyaya, and M. Mattson, "Seed location mapping using RTK GPS," *Trans.-American Society of Agricultural Engineers*, vol. 47, no. 3, pp. 909–914, 2004.

[4] K. Morgan, "A step towards an automatic tractor," *Farm mech*, vol. 13, no. 10, pp. 440–441, 1958.

[5] A. H. Göktoğan, S. Sukkarieh, M. Bryson, J. Randle, T. Lupton, and C. Hung, "A rotary-wing unmanned air vehicle for aquatic weed surveillance and management," *J. Intell. Robotics Syst.*, vol. 57, pp. 467–484, January 2010.

[6] V. Rusanov, "Effects of wheel and track traffic on the soil and on crop growth and yield," *Soil and Tillage Research*, vol. 19, no. 2-3, pp. 131 – 143, 1991.

[7] C. Phan and H. H. Liu, "A cooperative UAV/UGV platform for wildfire detection and fighting," in *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing*, (Beijing), pp. 494–498, Ieee, Oct. 2008.

[8] L. Chaimowicz, A. Cowley, D. Gomez-Ibanez, B. Grocholsky, M. Hsieh, H. Hsu, J. Keller, V. Kumar, R. Swaminathan, and C. Taylor, *Deploying air-ground multi-robot teams in urban environments*, vol. III, pp. 223–234. Springer, 2005.

[9] "RHEA: Robot Fleets for Highly Effective Agriculture and Forestry Management," *http://www.rhea-project.eu/*, accessed 08-Apr-2011.

[10] C. L. McCarthy, N. H. Hancock, and S. R. Raine, "Applied machine vision of plants: a review with implications for field deployment in automated farming operations," *Intelligent Service Robotics*, vol. 3, pp. 209–217, Aug. 2010.

[11] K. R. Thorp and L. F. Tian, "A review on remote sensing of weeds in agriculture," *Precision Agriculture*, vol. 5, no. 5, pp. 477–508, 2004.

[12] K. C. Swain, S. J. Thomson, and H. P. W. Jayasuriya, "Adoption of an unmanned helicopter for low-altitude remote sensing to estimate yield and total biomass of a rice crop," *Trans. of the ASABE*, vol. 53, pp. 21–27, Jan-Feb 2010.

[13] M. S. Moran, Y. Inoue, and E. M. Barnes, "Opportunities and limitations for image-based remote sensing in precision crop management," *Remote Sensing of Environment*, vol. 61, pp. 319–346, Sep 1997.

[14] D. W. Lamb and R. B. Brown, "Remote-sensing and mapping of weeds in crops," *Journal of Agricultural Engineering Research*, vol. 78, pp. 117–125, Feb 2001.

[15] R. B. Brown and S. D. Noble, "Site-specific weed management: sensing requirements - what do we need to see?," *Weed Science*, vol. 53, pp. 252–258, Mar-Apr 2005.

[16] M. Beckes and J. Jacobi, "Classification of weed patches in quickbird images: Verification by ground truth data," *EARSeL eProceedings*, vol. 5(2), pp. 173–179, 2006.

[17] J. A. J. Berni, P. J. Zarco-Tejada, L. Suarez, and E. Fereres, "Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, pp. 722–738, Mar 2009.

[18] R. Sugiura, N. Noguchi, and K. Ishii, "Remote-sensing technology for vegetation monitoring using an unmanned helicopter," *Biosystems Engineering*, vol. 90, pp. 369–379, Apr 2005.

[19] D. Wulfsohn, "Sampling techniques for plants and soil: In. advanced engineering systems for specialty crops: A review of precision agriculture for water, chemical, and nutrient application, and yield monitoring.," *Landbauforschung Völkenrode*, vol. Special Issue 340, pp. 3–30, 2010.

[20] D. Slaughter, D. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, pp. 63–78, Apr. 2008.

[21] a. Piron, F. Van Der Heijden, and M. Destain, "Weed detection in 3D images," *Precision Agriculture*, pp. 1–16, Nov. 2010.

[22] B. Huhle, T. Schairer, P. Jenke, and W. Straßer, "Fusion of range and color images for denoising and resolution enhancement with a non-local filter," *Comp. Vision and Image Understanding*, vol. 114, pp. 1336–1345, Dec. 2010.

[23] G. Alenya, B. Dellen, and C. Torras, "3D modelling of leaves from color and ToF data for robotized plant measuring," in *Proc. of the International Conference on Robotics and Automation*, (accepted), 2011.

[24] M. Kraft, N. Regina, S. a. D. Freitas, and A. Munack, "Test of a 3D Time of Flight Camera for Shape Measurements of Plants," in *CIGR Workshop on Image Analysis in Agriculture*, no. August, (Budapest), 2010.

[25] A. Nakarmi and L. Tang, "Inter-plant Spacing Sensing at Early Growth Stages Using a Time-of-Flight of Light Based 3D Vision Sensor," in *ASABE Meeting Presentation*, no. 1009216, 2010.

[26] M. Weis and M. Sökefeld, *Detection and Identification of Weeds*, ch. 8, pp. 119–134. Dordrecht: Springer Netherlands, 2010.

[27] M. Persson and B. Åstrand, "Classification of crops and weeds extracted by active shape models," *Biosys. Eng.*, vol. 100, pp. 484–497, Aug. 2008.

[28] M. R. Garey and D. S. Johnson, *Computers and Intractability*. W. H. Freeman and Co., 1979.

[29] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[30] K. Helsgaun, "General k-opt submoves for the Lin-Kernighan TSP heuristic," *Mathematical Programming Computation*, vol. 1, pp. 119–163, July 2009.

[31] E. K. Baker, "An exact algorithm for the time-constrained traveling salesman problem," *Operations Res.*, vol. 31, pp. 938–945, Apr. 1983.

[32] C. Walshaw, "A multilevel approach to the travelling salesman problem," *Operations Research*, vol. 50, pp. 862–877, Sept. 2002.

[33] B. P. Gerkey and M. J. Mataric, "Sold!: auction methods for multirobot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 758–768, Oct. 2002.

[34] L. E. Parker, "Alliance: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots," in *Proc. IROS*, pp. 776–783, 1994.

[35] R. M. Murray, "Recent Research in Cooperative Control of Multivehicle Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, 2007.

[36] G. M. Hoffmann, S. L. Wasl, and C. J. Tomlin, "Distributed cooperative search using information-theoretic costs for particle filters with quadrotor applications," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pp. 21–24, 2006.

[37] K. S. Alessandro Arsie and E. Frazzoli, "Efficient Routing Algorithms for Multiple Vehicles With no Explicit Communications," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2302–2317, 2009.

[38] L. Ziska and G. Runion, *Future weed, pest and disease problems for plants*, pp. 261–287. Boca Raton FL: CRC Press, 2007.

Paper B

An Autonomous Robotic System for Mapping Weeds in Fields

Karl Damkjær Hansen, Francisco Garcia-Ruiz, Wajahat Kazmi,
Morten Bisgaard, Anders la Cour-Harbo, Jesper Rasmussen,
Hans Jørgen Andersen

The paper is published in the
Proceedings of the 8th IFAC Symposium on Intelligent Autonomous Vehicles, 2013

Abstract

The ASETA project develops theory and methods for robotic agricultural systems. In ASETA, unmanned aircraft and unmanned ground vehicles are used to automate the task of identifying and treating weeds in sugar beet fields. The framework for a working automatic robotic weeding system is presented along with the implemented computer vision systems.

1 Introduction

The use of pesticides is detrimental to the environment. However, farmers must treat their fields against weed infestations to keep their business profitable. The current practice is to spray the entire field even if the weed distribution is heterogeneous. This herbicide discharge can be greatly reduced if the application is targeted only at actual infestations instead. However, it is required that the infestations are discovered and identified before they begin to compete with the crops. Practically, this is not possible if the weeds have to be surveyed by humans; this is simply too costly.

The ASETA project [1] is developing a system for autonomously mapping weeds in fields by means of robots, airborne and ground-based, fitted with advanced camera equipment. The airborne robots are based on small-scale helicopters that provide the system with multi-spectral aerial images. Using data from the helicopters, the system identifies infestations in a field and then dispatches autonomous ground vehicles to the infestations to exactly identify and localize the weeds.

In this paper, the framework and key technologies for integrating this system are described.

1.1 The ASETA Case

In their review of the current state of the art of robotic weed control systems, Slaughter et al. [2] report more than 50% yield loss if weeds are not controlled in row crops. They further note the problem that the weeds closest to the crops are the most harmful and that these are also the most difficult weeds to control. The consequence is that some fields must be hand-hoed, which is costly and inefficient. The aim of the ASETA project is to address this problem, and provide a solution for inexpensive, consistent, and reliable robotic weeding.

ASETA is working with a system of ground based and aerial vehicles. Both are unmanned and autonomous. Through a series of steps, the robots will identify and localize any weed infestations in a given field. The ASETA project works with a case of thistle (Cirsium arvense) infestations in sugar beet fields [3].

The topic of site-specific weed control is surveyed by Christensen et al. [4], where they classify the treatment of the fields in four levels:

1. Individual plant treatment
2. Treatment of grids (several plants)
3. Subfield treatment
4. Whole-field treatment

The ASETA project works in the first two levels, focusing on single plants and smaller patches.

A theoretical infrastructure of an agricultural decision support system for robotic site-specific weed management was proposed in [5]. That work takes a holistic approach and encompass everything needed to make such a system operational. In their terminology, the ASETA project focuses on the subsystem called the "current year decision system". This is concerned with the current state of a field and which treatment to apply to maximize the immediate yield.

1.2 Related Projects

Precision agriculture, the area of research on targeted treatment, has received much attention with the maturity of mobile robotics research and especially with the advent of cheap high-precision sensors such as GPS.

The RHEA project [6] was launched in 2010 with a mission much like ASETA's to reduce pesticide usage in agriculture. In RHEA, work is done in many area, but as shown in [5], they take a holistic approach and consider the entire precision agricultural system.

Projects like ASETA and RHEA rely on the incremental knowledge gained over the years in the fields of robotics, agriculture, and computer vision. In the intersection between agriculture and robotics, agricultural field robots, wheeled robots capable of carrying varying types of implements, have been researched and developed by several groups. Approaches range from fully-automated tractors [7] to dedicated platforms like the API [8], the HortiBot [9], and the BoniRob [10].

Much research has concerned the implements for autonomous robots. Usually such implements are combinations of vision systems and intelligent sprayers or hoeing tools. Examples of such developments are: Lettuce hoeing [11], date tree spraying [12], sugar beet hoeing [13]. Christensen et al. [4] and Slaughter et al. [2] provide good overviews of the technology.

2 Methods

2.1 Multiscale Imaging

The core idea in ASETA is to use a multiscale imaging approach. This entails taking aerial images at high altitudes, and then gradually lowering the altitude, to obtain images with higher resolution. At some point the ground vehicles will take over and perform imaging of individual plants.

A simplified process with a single unmanned aircraft system (UAS) and a single unmanned ground vehicle (UGV) looks like this:

- An operator defines the bounding polygon of a field.
- The UAS takes images of the field from high altitudes.
- The images are processed, indicating areas of interest for closer inspection.
- The UAS flies to the indicated ares and obtains higher resolution images (lower altitude), which are processed for indications of weed infestations.
- The UGV is dispatched to the areas that need attention.
- The UGV identifies the exact shape and position of each piece of weed and reports it to the system.
- The process continues until all weeds in the field have been mapped.

This multiscale imaging approach saves time because it quickly directs the UGVs to actual infestations. It also has the advantage of acquiring overview images of the entire field in the process. These images can be used for weed estimates, in the locations, where the UGVs do not go. However, aerial images are not as precise as ground-based images. Chistensen et al. [4] deems aerial based sensing fitting for the two coarsest classifications of site-specific treatment (sub-field and whole-field). They, however, did not have UASs in mind, but rather piloted aircraft and satellites; UAS imaging provides even higher resolutions. By using ground-based sensing to correct and calibrate the aerial measurements, the aerial images may be used at the next levels (several plants and single-plant).

2.2 System Architecture

The conceptual structure of a system with one UGV and one UAS is shown in Fig. B.1. It consists of a task manager that automatically decomposes the task of the entire system (i.e. to survey a given field) into tasks for the individual subsystems. The supervisors interpret the tasks and command the vehicles to move to the indicated positions, while

they keep track of the execution. The data from the vehicles are processed to update the map of the field in the database. When such updates occur, the task manager factors the new information into the plans and changes the tasks of the vehicles accordingly.

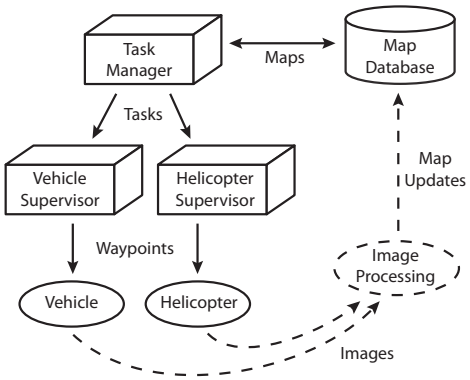


Fig. B.1: The conceptual system of ASETA. The task manager creates tasks for the vehicles. The tasks are handled by the supervisors, which monitor and provide the vehicles with waypoints in the correct sequence. The data obtained by the vehicles are processed to provide updates to the maps in the database.

The description of the simplified process above (Sec. 2.1) is static, in the sense that information only travels from the UAS to the UGV. The system proposed here follows the same general idea but is more dynamic because it allows the vehicles to work in parallel and cooperate.

A short description of the components is given below.

Task Manager

The job of the task manager is to decompose the overall task in such a way that the UGVs and the UASs perform the execution in the fastest and least resource demanding fashion.

In essence, what the task manager has to do is solve a job shop scheduling problem. Because the abilities of different vehicles are overlapping. E.g. the UAS can quickly photograph the entire field, albeit at a low resolution. This is also possible for the UGV; it will take a long time, but provide a high resolution. So the job is to figure out which vehicles to use to acquire which images.

Further, the images must be taken at different locations, so the execution time of a task is not only dependent on the time it takes to take the photograph, but also the travel time between the locations. This alludes to a case of the traveling salesman problem.

So, the task manager is tasked with two cases of combinatorial optimization. The approach of the ASETA project is to solve these using genetic algorithms (see section 2.3).

Supervisors

The supervisor interprets the tasks given by the task manager and provides the vehicles with lower-level commands such as waypoints, reference trajectories or when to take an image. The primary task of the supervisors is to ensure that the tasks that are passed on to the vehicles are executable, but they also function as a standardized interface between the task manager and the vehicle. This way, the task manager can ignore the dynamics of the vehicles when planning.

Vehicle and Helicopter

In the ASETA case, the UAS is based on a small-scale helicopter (Fig. B.2) and the UGV is based on a four wheeled robotic platform (Fig. B.3). But, the general system allows for several, possibly different vehicles. In this way, the system can be suitable for a range of different scenarios.

Fig. B.2: The UAS, based on a Maxi Joker 3 RC-helicopter, equipped with a multi spectral camera mounted in a gimbal device (front), a mini-ITX computer (underside), and IMU and GPS (on tail).

Image Processing

The automatic processing of the image data provided by the vehicles is essential to the system. The ASETA project does work in both ground-based and aerial imaging. The aerial image processing focus on determining weed patches and and the ground-based image processing works with the identification of single plants. These two topics are described in sections 2.5 and 2.6.

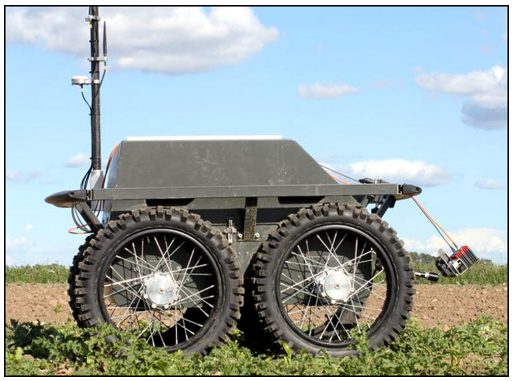


Fig. B.3: The UGV is based on a the RobuROC4 platform from Robusoft. It is a skid-steered, four-wheel driven vehicle with custom onboard computer, sensor suite, and camera setup.

Map Database

The end product of the entire automation exercise is to build a map of the field, indicating spots of weeds. Initially, the database will hold only the outline of the field, and as the vehicles provide more information, the maps will be populated.

The ASETA project focus on mapping thistles in sugar beet fields, but having several other image processing algorithms and sensors on the vehicles could enable the system to produce several different maps in the same run. These maps could include soil nutrition levels, plant growth stages, pest infestations as well as the weed map.

2.3 Automatic Planning

The goal of the system is to have a complete survey of a given field. The automatic planner decomposes this goal into several states, each composed of a location and an action that the vehicle must take in that location. The vehicles must visit these locations in a sequence. So in order to save fuel and time, the planner must find the shortest path between the coordinates; this is a case of the traveling salesman's problem.

The planning is done with a genetic algorithm (GA). The GA used for the planing is based on the path-representation described in [14], and uses the four mutations: Displacement, exchange, inversion, and insertion [15–17].This GA does not guarantee to find the optimal solution nor to converge to it, however it will often converge on good candidate solutions. The difference between the candidate and the optimal solutions is tolerated, because the environment and vehicle dynamics incur so much uncertainty that it is unknown whether the optimal solution in terms of distance is in fact the best.

Although the GA might not converge to the optimal solution, it must be given some computation time to arrive at whatever near-optimal solution it is converging to. It is usually up to the designer of the algorithm to decide on how much time the algorithm is given, which is not always easy at design time as the runtime increases with problem size and is dependent on the quality of the initial guess. A contribution from the ASETA project is an adaptive stopping criterion for GAs, that indicates when the algorithm has reached an acceptably good solution [18].

The distance that the robots travel depends on the plan that the GA constructs. A simple measure of the solution quality is the euclidean distance between the points. This is easily computable but introduces sharp turns that are not realizable by the vehicles because of kinematic and dynamic constraints.

Currently, an alternative to the Euclidean distance measure is being studied under the ASETA project. It bases the distance measure on Dubins curves, which are composed of line and arc segments. These curves are differentiable and match the vehicle dynamics better, but they also introduce more computations as well as a continuous variable in the heading of each waypoint. These two problems are addressed by using the Euclidean measure first, and later in the process when the solutions are converging substitute the distance measures with Dubins measures for the relevant arcs (i.e. the candidate arcs for the final solution). This work is based on the alternating algorithm of Savla et al. [19] with the considerations of headings introduced by Ny et al. [20]. The notations and results of [21] are used to achieve efficient computations of the Dubins curves.

The algorithm is relying on the simple cost function of distance traveled by the robots. A further step could be to use a cost function like energy usage, which could improve the system performance as the vehicles are battery-powered.

The algorithm is currently static, in the sense that if a new goal is set, then the algorithm must be restarted. Work is being done to dynamically inject new goals into the running algorithm as well as deleting fulfilled goals.

2.4 Path Planning

The path planning for the UGV is done in 2D assuming the field to be a flat surface. The grid layout of the field is assumed to be known, without knowing the exact position of every plant. The planner uses Dubins' curves to smooth the transitions between the edges of the field. Recall that the UGV is skid-steered and able to turn on the spot, however doing so will churn up the soil and possibly destroy crops. Using Dubins' curves is a compromise between churning up the soil and diverging from the desired path. Figure B.4 shows an example of a UGV visiting a number of goals in a field.

Currently, no emphasis is given to unknown obstacles in the field, such as rocks, other vehicles, or humans. Static obstacles can be programmed into the field layout, but otherwise it is up to the operators to intervene if emergency situations occur. As

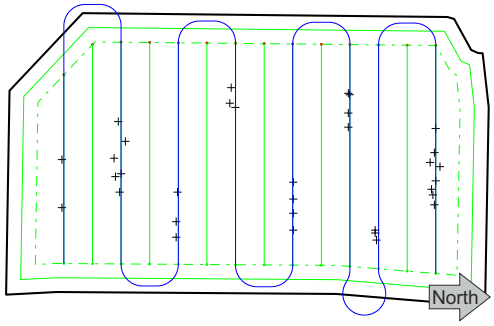


Fig. B.4: Trajectory of a UGV visiting hotspots in a field (57.0144 N, 9.993 E). Plusses (+) indicate goals, greeen lines indicate paths in the field, and the blue line indicates the planned path of the UGV. The straight segments are joined with Dubins curves.

this is an experimental system and emphasis is on the overall functionality of the system, such robustness is not top priority.

2.5 Aerial Image Processing

The aerial image processing is based on color analysis. The fact that different plant species show different colors is used to discriminate between them. Physiological internal processes determine the important features for discrimination (i.e. chlorophyll absorption bands, red edge inflection point) and can be detected by narrow band multispectral imaging with a sufficient spatial resolution. Hence, it is possible to distinguish different plants by identifying the spectral features where they show the maximum difference [22, 23]. In the ASETA project an extensive survey of the spectra of thistles and sugar beet leaves has been conducted under real life conditions, Fig. B.5 shows the average spectra of the two species.

One of the first objectives of the ASETA was to investigate the possibility of discriminating sugar beets and thistles based on their spectra under field conditions. Principal component analysis was used to assess the separability of sugar beets and thistles, and determine the most prominent features (wavebands) that show higher variability. First results show a great separation when using uncorrelated variables and indicated wavebands centered at 550, 680, 750 and 940 nm as the most significative when classifying those two species (see Fig. B.6).

The aerial vision system used in this research is based on a multispectral narrow-band filter camera (seen in Fig. B.2). The MiniMCA-6 (Tetracam Inc., USA) weighs 695 g and consists of six individual digital cameras arranged in a 3 by 2 array and synchronized so they can be triggered at the same time. Each of the cameras is equipped with a 1.3

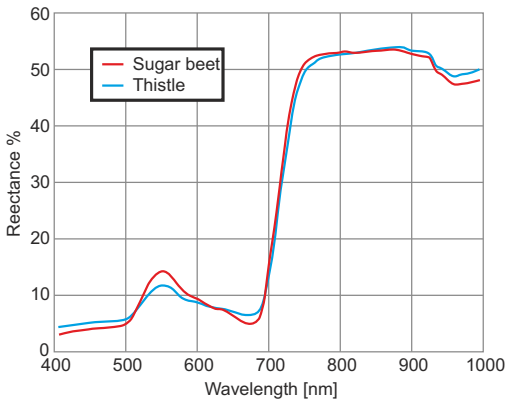


Fig. B.5: Spectral signature of sugar beet (red) and *Cirsium Arvense* L./thistle (blue) in the visual-NIR range. Significant difference is seen around 550, 680, and 940 nm.

megapixel CMOS sensor. Interchangeable narrow bandpass filters are placed in front of the optics to block the unwanted frequencies. The configuration of filters (see Table B.1) is selected to coincide with the wavebands where the main physiological phenomena are reflected as well as to allow the calculation of the most important vegetation indices. For technical reasons, the filters does not match the frequencies identified in Fig. B.5 exactly, but are close enough for identification purposes.

Table B.1: The filters mounted on the Mini MCA camera

	Mini MCA Filters					
Wavelength*	488	550	610	675	780	950
Bandwidth*	10	10	10	10	10	40

* nanometers

The multiscale imaging process decribed in section 2.1 is used for quickly gathering information with a low resolution (approximately 50 mm/pixel), flying at high altitudes, and for a finer detection at plant level using higher resolution images (10–20 mm/pixel) taken at low altitudes.

First, a coarse vegetation map is generated from lower resolution images using the excess green index (ExG) which is highly effective in masking out the green objects from the bare soil background [24]. A close relation exists between the mean ExG for a certain area and its vegetation density, and comparisons can be made within the same

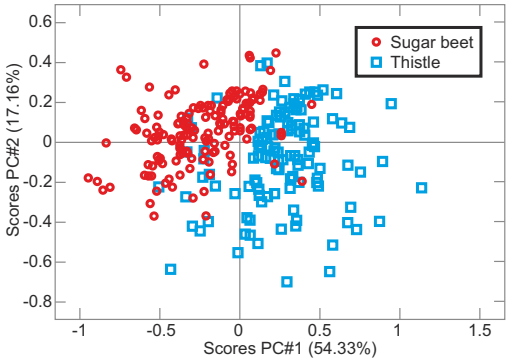


Fig. B.6: Principal Component Score plot of dataset composed by 80 samples of sugar beet (red) and 80 samples of thistle (blue) plants. PC1 and PC2 accounting for the 71.5 % of the variability.

image. Regions with high biomass are ranked by importance (size and density) and geo-positioned. The traditional ExG is computed using the red, green, and blue (RGB) channels of an ordinary digital color camera. In this multispectral approach, the three channels best corresponding to RGB are used (675, 550, and 488 nm).

The information is sent to the task manager (see section 2.2) to proceed to a closer inspection for crop-weed discrimination. At low altitudes the spatial resolution increases, and the spectral mixing decreases yielding a high amount of pure pixels per plant. The crop rows are clearly seen and detected, and plants in between the rows are classified as thistles due to their position . Once classified, a library can be made with the spectral endmembers collected from the purest inter-row thistle pixels. The intra-row plants are matched with the now known endmembers and labeled as sugar beets or thistles to produce an aerial 2D weed map.

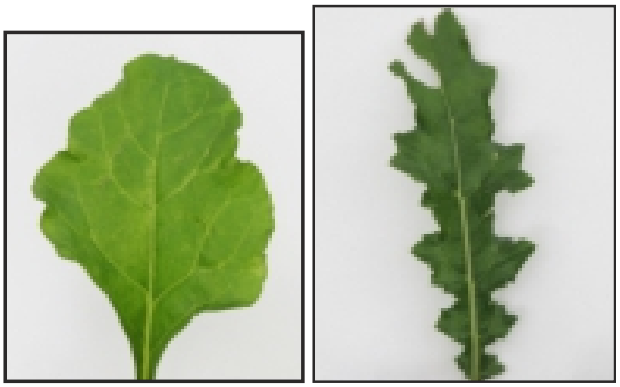
A continuous feedback is established with the UGV, which is making a more detailed characterization and estimation of weed density. This allows an online update of the relationship of mean ExG value versus plant density for the coarse aerial imaging and a supervisory update of the classification at the finer stage, which will improve the weed map even after the aerial images were taken.

2.6 Ground vision

The analysis of aerial imagery prompts the ground vehicle for a closer inspection. In the ground based image processing, the green color of vegetation is again a first step since greenness of the plants' leaves distinguishes them well against the background soil. However, the ability to resolve overlapping leaves is limited with 2D vision. Fig. B.8 shows two segments of an image. In (a), an overlapping thistle leaf has occluded the

sugar beet leaf. This hides some of the shape features of the sugar beet leaf and the leaf should therefore be marked as unfit for species classification. In the (b) segment, the leaf is not occluded and it is better suited for species classification.

To detect occlusions 3D imaging is crucial. Using the depth information, detecting occlusions becomes trivial. Once a leaf has been qualified for classification, either 2D or 3D imaging can be used for the further processing.



(a) Sugar Beet Leaf **(b)** Thistle Leaf

Fig. B.7: Comparison of leaf shapes and color of target species.

For 3D imaging, stereovision is a commonly used technique, but it suffers from correspondence and efficiency problems for close range leaf imaging. For this reason, stereovision imaging is largely limited to indoor well-lit conditions, or to an overall plant canopy measurement in outdoor conditions [25].

To overcome these limitations, we are using Time of Flight (ToF) cameras along with a color camera (Fig. B.9) to achieve a closer depth analysis inside plant canopies. ToF cameras are active sensors working in the Near Infrared (NIR) region. They emit infrared light and measure distances to the objects in the view based on the time it takes for the light to return.

Because the ToF cameras use NIR light, the reflectance-transmittance characteristics of the leaf surface in this spectrum must be taken into account. Any incident light is partly reflected from the leaf surface, partly absorbed and the rest is transmitted through the leaf, but only the reflected portion is interesting in ToF imaging. This topic has received a great deal of research. Jacquemoud and Baret [26] proposed a Reflectance-Transmittance model for green leaves which show about 51% reflectance, 45% transmittance and 4% absorption for green soyabean leaves in the NIR region. Indeed, the model shows that the NIR region provides highest possible reflectance (just under 50 %) of the frequencies in the visible-NIR spectrum in the frequency band between 700 nm and 1300 nm. This fits with our observations of sugar beet and thistles

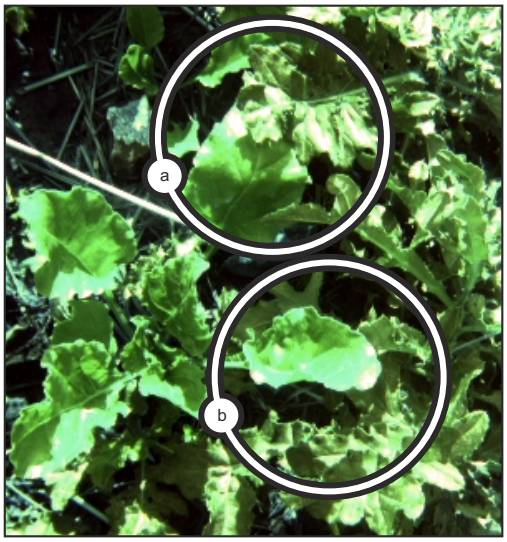


Fig. B.8: Complex overlapping scenario. Thistle occluding sugar beet (a) and sugar beet without occlusion (b).

shown in Fig B.5. ToF cameras operating at 850 nm are hence quite suitable for plant imaging. Further, they produce depth data at more than 30 fps, while not having the correspondence or efficiency problems of stereovision.

However, ToF cameras have their own shortcomings. Other than their low resolution sensors (200x200 max), one major problem is their saturation under sunlight. In ToF cameras, integration time (IT) controls the duration for which the incoming signal is integrated onto the imaging sensor. IT must be high enough to allow sufficient depth estimation but less than the saturation threshold. The gap between these two boundaries of operation depend on the ambient light and it becomes very narrow under sunlight. It is one of the research points of the ASETA project to look into the usefulness of ToF under these conditions.

Fig. B.10 shows depth data of a single leaf under room and sunlight conditions. The graphs show the variation of depth data of two individual pixels averaged across several frames, as well as the average of their 20x20 pixel neighborhood. The pixels were located on the surface of the leaf. The point where the depth data of the pixels and the average of the neighborhood starts getting out of sync; the data is no longer reliable. This fact can render ToF cameras useless unless a shade is used to cast shadow on the view. Kazmi et al. [27] presents a detailed analysis about leaf imaging with ToF cameras under sunlight, shadow and room conditions.

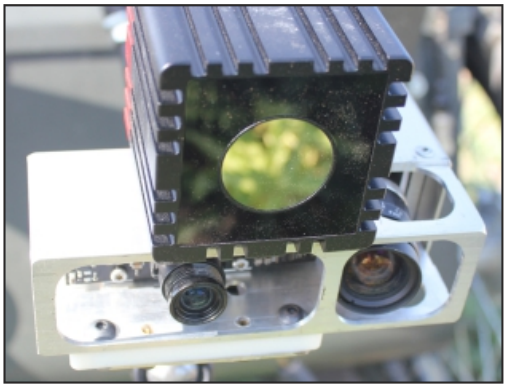


Fig. B.9: Cameras mounted on Ground Vehicle. Swissranger SR4000 time-of-flight camera (top), PMD CamBoard time-of-flight camera (bottom left), and Point Grey Flea RGB camera (bottom right).

In order to classify a plant as either thistle or sugar beet, the approach is to use shape-features of the leaves (see Figs: B.7a, B.7b) along with their relative greenness. To extract shape boundaries, an algorithm using triangular decomposition of the image similar to [28] is being developed. It is a generic algorithm which estimates salient regions from the edges of an object. The color channel from the color camera is mapped onto the ToF data using stereo-calibration of the two cameras. When a plant is classified, a feedback will be generated to update the weed map from aerial imagery.

2.7 Cooperation

In terms of solving the basic tasks for a crop and weed management system, the automatic planning is capable of producing the necessary waypoints for this task. However, the ASETA project attempts to push the intelligence of such a system further than basic automation and a large part of that is the cooperation between robots. One of the things being researched is how robots can cooperate with very limited communication between them. An example where this could be relevant is a farmer that has purchased a simple UAS to do mapping of his fields. After some time he purchases a newer more advanced UAS and would like them to help each other mapping the fields. The simple UAS is unable to do cooperation, but the newer UAS is capable of assisting with the mapping, simply by estimating the intentions of the older UAS from, say, ADS-B beacon data.

Currently the research focuses on how to provide a quantitative estimate of the intentions from simple beacon information. This is done using model based baysian filtering with a short and a long term prediction. As system model, a probability field mapping is used to assign probabilities to the individual waypoints depending on where

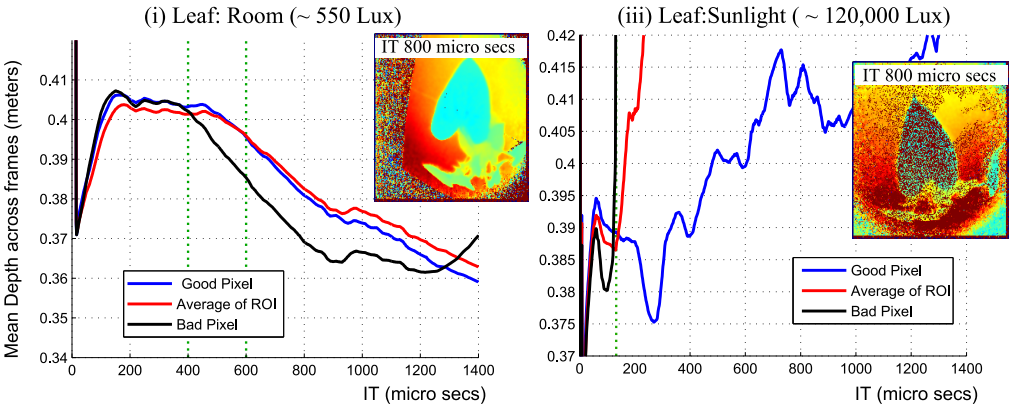


Fig. B.10: Comparison of leaf imaging with PMD CamBoard ToF camera under Room and Sunlight. Two sample images at IT 800 ms illustrate the difference.

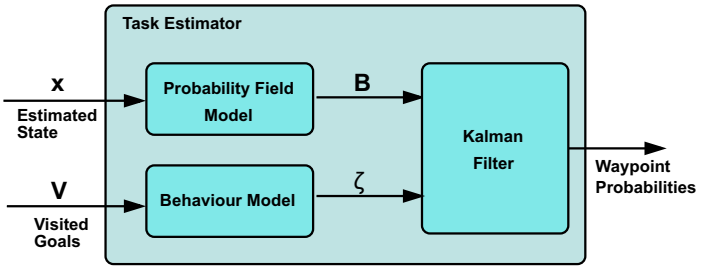


Fig. B.11: Waypoint probability estimator.

they are located in relation to the motion of the helicopter. It is constructed as a dynamic function of two parts; a part called probability field map, which increases the probability for waypoints near the expected future positions of the helicopter, and a part called a dissipation function that gradually reduces the probability of all waypoints such that the waypoints need to stay within the future path of the helicopter to maintain a high probability.

The measurement model takes the information of which waypoints the agent previously have visited and from this attempts to predict the probability for each waypoint. This probability is then used as the measurement for the kalman filter which results in the measurement model matrix being the identity matrix. This prediction is done through a set of behavioral models, each modeled mathematically based on a set of assumptions. Examples of such behavioral models could be different flight patterns like a “lawnmover” pattern, a spiral pattern or a nearest neighbor pattern.

3 Discussion

While aerial imaging has been used for several years in production farming, the new platforms, brings about a range of new possibilities. Traditional aerial imaging is done using piloted planes, and the images are orthorectified, which is a rather time-consuming and expensive process that must be planned well ahead and is dependent on good weather conditions. The aerial robotic platforms provides the images whenever the farmer needs them with less dependence on weather conditions and external planning.

The ASETA project shows that it is plausible to use mobile robotics in future weed mapping and targeted weed control. At the time of writing, 8 large-scale test campaigns has been conducted in order to obtain real life images and measurements, and test the robotic platforms outside of lab-conditions. We have found that the data obtained so far in the campaigns is usable in the agricultural analyses.

Presently, several steps towards a functional system have been taken, although there is still limitations:

The UAS is a product of several years of work in the UAV lab Aalborg University. It is capable of autonomous flight between waypoints given by the user. It performs well even in the presence of strong wind and changing GPS-conditions and has proven to be a reliable platform for aerial imaging.

The planning algorithm is capable of solving traveling salesman problems (TSP) for a single UAS and a single UGV individually, it is still an open problem to solve the multiagent heterogeneous TSP of the two or possibly more robots.

The aerial image analysis is currently able to identify areas of high bio-mass density, thus indicating areas with weed infestations. Next step is to identify weeds based on spectral analysis.

The work on the ground vision has focused on the identification of leaf shapes and the usability of ToF cameras in outdoor conditions. Presently, the analysis will characterize the leaf edges in a well conditioned RGB image, e.g. low background noise. Next step will be to apply the analysis to 3D images and observe the differences and advantages compared to 2D imaging.

The ASETA project will continue to integrate the fields of agricultural image analysis and robotics, and will demonstrate a working system in 2013.

4 Acknowledgment

This work is supported by the Danish Council for Strategic Research under grant no. 09-067027 (ASETA). See `www.aseta.dk` for more details.

References

[1] A. la Cour-Harbo, "ASETA Project - Adaptive Surveying and Early treatment of crops with a Team of Autonomous vehicles," 2010. [Online]. Available: http://www.aseta.dk

[2] D. C. Slaughter, D. K. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, no. 1, pp. 63–78, Apr. 2008.

[3] W. Kazmi, M. Bisgaard, K. D. Hansen, F. Garcia-Ruiz, and A. la Cour-Harbo, "Adaptive Surveying and Early Treatment of Crops with a Team of Autonomous Vehicles," in *Proceedings of the 5th European Conference on Mobile Robotics*, 2011, pp. 253–258.

[4] S. Christensen, H. T. Søgaard, P. Kudsk, M. Nørremark, I. Lund, E. S. Nadimi, and R. Jørgensen, "Site-specific weed control technologies," *Weed Research*, vol. 49, no. 3, pp. 233–241, Jun. 2009.

[5] C. Fernández-Quintanilla, J. Drorado, C. S. Martín, J. Conesa-Muñoz, and A. Ribeiro, "A Five-Step Approach for Planning a Robotic Site-Specific Weed Management Program for Winter Wheat," in *Proc. 1st International Workshop on Robotics and Associated High Technologies and Equipment for Agriculture.* Madrid: Producción Gráfica Multimedia, PGM, 2011, pp. 3–12. [Online]. Available: www.rhea-project.eu

[6] P. Gonzalez-de Santos, "RHEA Project - Robot Fleets for Highly Effective Agriculture and Forestry Management," 2010. [Online]. Available: http://www.rhea-project.eu/

[7] B. S. Blackmore, H. W. Griepentrog, H. Nielsen, M. Nørremark, and J. Resting-Jeppesen, "Development of a deterministic autonomous tractor," in *Proceedings CIGR*, vol. 11, 2004, p. 2004.

[8] T. Bak and H. Jakobsen, "Agricultural robotic platform with four wheel steering for weed detection," *Biosystems Engineering*, vol. 87, no. 2, pp. 125–136, 2004.

[9] R. N. Jørgensen, C. G. Sørensen, J. Maagaard, I. Havn, K. Jensen, H. T. Søgaard, and L. B. Sørensen, "Hortibot: A system design of a robotic tool carrier for high-tech plant nursing," *Agricultural Engineering International: the CIGR Ejournal*, 2007.

[10] A. Ruckelshausen, P. Biber, M. Dorna, H. Gremmes, R. Klose, A. Linz, F. Rahe, R. Resch, M. Thiel, D. Trautz *et al.*, "Bonirob–an autonomous field robot platform for individual plant phenotyping," in *Proceedings of the 7th European Conference on Precision Agriculture, Wageningen Academic Publishers*, 2009, pp. 841–847.

[11] J. Heraud, "Blue River Technologies," 2011. [Online]. Available: http://www.bluerivert.com

[12] A. Shapiro, E. Korkidi, A. Demri, O. Ben-Shahar, R. Riemer, and Y. Edan, "Toward elevated agrobotics: Development of a scaled-down prototype for visually guided date palm tree sprayer," *Journal of Field Robotics*, vol. 26, no. 6-7, pp. 572–590, Jun. 2009.

[13] J. Rasmussen, H. W. Griepentrog, J. Nielsen, and C. B. Henriksen, "Automated intelligent rotor tine cultivation and punch planting to improve the selectivity of mechanical intra-row weed control," *Weed Research*, vol. 52, no. 4, pp. 327–337, Aug. 2012.

[14] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[15] D. B. Fogel, "Applying Evolutionary Programming to Selected Traveling Salesman Problems," *Cybernetics and Systems*, vol. 24, no. 1, pp. 27–36, Jan. 1993.

[16] W. Banzhaf, "The "molecular" traveling salesman," *Biological Cybernetics*, vol. 64, no. 1, pp. 7–14, Nov. 1990.

[17] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin-Heidelberg: Springer-Verlag, 1999.

[18] K. D. Hansen and A. la Cour-Harbo, "An Adaptive Termination Criterion for Genetic Algorithms in Travelling Salesman Problems," 2013, submitted.

[19] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins' vehicle," in *Proceedings of the American Control Conference, 2005.* Portland, OR, USA: IEEE, 2005, pp. 786–791.

[20] J. L. Ny, E. Feron, and E. Frazzoli, "On the Dubins Traveling Salesman Problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2012.

[21] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, Mar. 2001.

[22] E. Vrindts, J. De Baerdemaeker, E. Meyer George, and J. DeShazer, "Optical weed detection and evaluation using reflection measurements," *SPIE proceedings series, International Society for Optical Engineering, Precision agriculture and biological quality*, vol. 3543, pp. 279–289, 1998.

[23] E. Vrindts, J. De Baerdemaeker, and H. Ramon, "Weed detection using canopy reflection," *Precision Agriculture*, vol. 3, no. 1, pp. 63–80, 2002.

[24] G. Meyer, J. Camargo Neto, D. D. Jones, and T. W. Hindman, "Intensified fuzzy clusters for classifying plant, soil, and residue regions of interest from color images," *Computers and Electronics in Agriculture*, vol. 42, no. 3, pp. 161–180, Mar. 2004.

[25] C. L. McCarthy, N. H. Hancock, and S. R. Raine, "Applied machine vision of plants: a review with implications for field deployment in automated farming operations," *Intelligent Service Robotics*, vol. 3, no. 4, pp. 209–217, Aug. 2010.

[26] S. Jacquemoud and F. Baret, "PROSPECT: A model of leaf optical properties spectra," *Remote sensing of environment*, vol. 34, no. 2, pp. 75–91, Nov. 1990.

[27] W. Kazmi, S. Foix, and G. Alenya, "Plant Leaf Imaging using Time of Flight Camera under Sunlight, Shadow and Room Conditions," *International Symposium on Robotic and Sensors Environments, IEEE*, p. (accepted for publication), Nov. 2012.

[28] R. Distasi and M. Nappi, "Image compression by B-tree triangular coding," *IEEE Transactions on Communications*, vol. 45, no. 9, pp. 1095–1100, 1997.

Paper C

Characterization of Genetic Algorithm Mutation Operators for Solving Traveling Salesman's Problem

Karl Damkjær Hansen, Anders la Cour-Harbo

The paper is submitted to be included in the
Proceedings of the Genetic and Evolutionary Computation Conference 2014, GECCO'14

Abstract

When using local search algorithms like genetic algorithms it is desirable to have both an fast and diverse search such as to increase the probability of a good result in a relatively short time. However, these properties are typically opposite and a good trade off can be difficult to find. In this work we examine three specific neighborhood functions for genetic algorithms to determine the entropy rate of each, these are the exchange, displace, and inversion mutations. The analysis is done in the context of solving asymmetric traveling salesman's problems using a path representation. The result allows for a balanced use of mutation operators such as to control population diversity during search.

1 Introduction

The traveling salesman's problem is an NP-hard combinatorial problem. Meta-heuristic algorithms can be used to obtain "good" solutions for the problem. Meta-heuristics like steepest ascend, simulated annealing and genetic algorithms rely on neighborhood functions to generate "neighboring" solutions from existing solutions, i.e. functions that modify existing solutions slightly. This is used by the meta-heuristic algorithm to explore the regions around known solutions for better solutions. There is no hard and fast rule as to how big the neighborhood can be. That is, the existing solution may be modified ever so slightly, brutally or anything in between. What is most desirable depends on the problem at hand and the ideas of the designer. In the following we will be characterizing a set of neighboring functions in the context of solving the traveling salesman's problem using genetic algorithms.

Genetic algorithms are modeled on Darwin's evolution theories. The algorithm keeps a population of solutions to the problem to be solved, the best of these solutions, the ones with the lowest cost, have a higher chance of producing offspring by being subjected to the neighborhood functions. As the generations pass, better and better solutions are found, until the algorithm is terminated.

In the genetic algorithm, the neighborhood functions are called mutations as a reference to the genetic makeup of the parents being mutated. Another flavor of neighborhood function where the genes of two parents are combined are called crossover functions. In the following, only mutations are considered.

When solving traveling salesman's problems (TSPs) with genetic algorithms, the path representation is a common and intuitive genome representation. The problem consists of a numbered list of coordinates, the genome is represented as a list of indices, each index referring to a set of coordinates in the problem. Thus, the genome

$$(1, 3, 4, 2, 5)$$

will direct the traveling salesman to travel from the first city (or set of coordinates) to the third, next to the fourth, over the second and to the fifth.

This representation has the property that any permutation of the trivial solution,

$$(1, 2, \cdots, n)$$

is feasible because every city has been visited exactly once. So, as long as the mutation operators working on the genome is only permuting it, every operation will result in feasible solutions.

Several mutation operators for the path representation has been developed. See [1] for a nice review of mutation and crossover operators for TSPs. In this paper, we will be looking specifically at three common mutation operators; the exchange, displace, and inversion mutations. These three mutations are easy to understand and their effects on the TSP are intuitive, further Larrañaga et al. note that these mutations are among the best performing mutations for the path representation [1].

Genetic algorithms are based on a population of genomes that are constantly mutated and bred. To explore the entire search space, it is desirable to have a diverse population. Conversely, if the desire is to exploit a smaller region of the search space, a more homogeneous population is needed. Maekawa et al. [2] and Tsujimura and Gen [3] have proposed entropy based methods to evaluate the diversity of the population. Tsujimura and Gen presents a locus measure, where the positions of the alleles in the genome is prioritized. Maekawa et al., on the other hand, prioritizes the adjacencies between the genomes. In this work, we make use of the adjacency method, as the "sequence of indices" formulation for the TSP problem may encode similar tours for different permutations of the genomes, specifically rotations:

$$(1, 2, 3, 4, 5) = (3, 4, 5, 1, 2)$$

1.1 Mutation Algorithms

The exchange mutation randomly chooses two entries to swap. See Algorithm 1.

Algorithm 1 The exchange mutation algorithm swaps two of the entries in a genome.

Require: original genome g_o
  procedure EXCHANGE MUTATION(g_o)
    $n \leftarrow \text{Length}(g_o)$
    $i_1 \leftarrow \text{RandomInteger}(1, n)$
    $i_2 \leftarrow \text{RandomInteger}(1, n) \neq i_1$
    $g_m \leftarrow g_o$
    $g_m[i_1] \leftarrow g_o[i_2]$
    $g_m[i_2] \leftarrow g_o[i_1]$
    return mutated genome g_m
  end procedure

The inversion mutation works mostly like the exchange mutation. However, the entries between the selected indices are also swapped, this equates to reversing the direction of travel for that section of the TSP. See Algorithm 2.

Algorithm 2 The inversion mutation algorithm reverses the direction of travel in a subsection of a genome.

Require: original genome g_o
  procedure EXCHANGE MUTATION(g_o)
    $n \leftarrow \text{Length}(g_o)$
    $i_1 \leftarrow \text{RandomInteger}(1, n)$
    $i_2 \leftarrow \text{RandomInteger}(1, n) \neq i_1$
    $g_m[i_1 : i_2] \leftarrow g_o[i_2 : i_1]$
    return mutated genome g_m
  end procedure

The displacement mutation selects a section two indices and "slides" that section. See Alg. 3. Another way of looking at it is that two sections are selected and swapped.

Algorithm 3 The displacement mutation algorithm moves a subsection of a genome from one position to a new.

Require: original genome g_o
  procedure EXCHANGE MUTATION(g_o)
    $n \leftarrow \text{Length}(g_o)$
    $i_1 \leftarrow \text{RandomInteger}(1, n)$
    $i_2 \leftarrow \text{RandomInteger}(1, n) > i_1$
    $i_3 \leftarrow \text{RandomInteger}(1, n) \geq i_2$
    $g_m \leftarrow g_o$
    $p \leftarrow i_3 - i_2$
    $g_m[i_1 : (i_1 + p)] = g_o[i_2 : i_3]$
    $q \leftarrow i_2 - i_1$
    $g_m[(i_1 + p + 1) : (i_1 + p + q)] = g_o[i_1 : (i_2 - 1)]$
    return mutated genome g_m
  end procedure

1.2 Proofs of effectiveness

If one mutation is to be used in a genetic algorithm, it is necessary that it can transform any genome into any other genome in a finite number of iterations. Otherwise, the algorithm will be unable to evaluate the entire search space. This may be acceptable

in algorithms using crossover operators or several mutation operators, but not for an algorithm relying only on a single mutation operator. Here we will prove that the three investigated mutations can in fact explore the entire search space.

We will use the same approach for all the proofs. We will start out with a random permutation of the trivial sequence; then, through a series of steps, transform this genome into the trivial sequence. If this is possible then the inverse series of operations are also possible, thus any random sequence must be transformable to any other random sequence.

Using the exchange mutation, we can simply start from the first entry in the genome, $g[1]$, then find the entry that has the value 1, $g[n]$, then do the same for $g[2]$ and so on until we reach $g[n-1]$, see Algorithm 4. Note that this notation receives the mutation parameters as inputs rather than via the RandomInteger operators in Algorithms 1, 2, and 3.

Algorithm 4 Transforming any random genome into the trivial sequence.

Require: genome g
  $n \leftarrow \text{Length}(g)$
  for $i \leftarrow 1, n-1$ **do**
    $j \xleftarrow{\text{index}} \text{Find}(i)$
    $\text{Mutation}(g, i, j)$
  end for

The same approach works for the inversion mutation, as the modification of the entries between the selected indices are irrelevant because the sequence between 1 and i is untouched. And, finally, by choosing the i_3 equal to i_2 in the displacement mutation the approach in Algorithm 4 works as expected, by "sliding" a single index sequentially into place.

The ability to explore the search space is only a guarantee that no possible solution is left unevaluated if the genetic algorithm is given infinite computation time. The operators should have the ability to both explore the search space as well as exploit promising search regions. The exploitation is made possible by the selection process of the genetic algorithm as it selects the most fit individuals as the basis for the next generation. If the mutation, however, completely mangles the genome so that it is no longer in the search region of the parent, no exploitation is achieved. The question is then how apt the mutation operators are to either explore or exploit.

In the following, we will present a method to test the exploration/exploitation ratio of the mutations using an entropy-based measure. Lastly we will present the results of this test for the three mutations applied to a set of standard TSP problems.

2 Methods

A measure of how exploratory/exploitive the mutation operators are may be defined in the amount of information that is being preserved between the original and the mutated individual. The more information that is preserved the less the individual has changed, and the less exploration is achieved. Using this definition, entropy rate can be used as a measure of the exploratory/exploitive nature of the operators. We will call this measure the exploration degree of the mutation.

2.1 Entropy Rate

The entropy rate H of a stochastic process X is defined as

$$H(X) = -\sum_i p_i(X) \log p_i(X) \,, \tag{C.1}$$

where p_i is the probability for the i^{th} outcome of the process.

If we regard the genome g as the stochastic process, we could analyze it by evaluating its transition matrix. However, if g has N entries, there are $N!$ permutations and the transition matrix will hold $(N!)^2$ entries, which is unmanageably big for even moderate sized genomes, and rules out numerical evaluations. Instead we use the adjacency matrix to characterize the process.

The adjacency matrix A defines which vertices of the problem are connected with an edge. For an edge going from vertex i to vertex j, the entry A_{ij} is 1, if there is no connection the entry is 0. E.g., the genome for an asymmetric tour:

$$(1,\, 3,\, 4,\, 2,\, 5)$$

has the corresponding adjacency matrix:

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The adjacency matrix has the nice property of being invariant to rotations of the entries in the genome. So the matrix above is the same for both the genome above and the genome:

$$(4,\, 2,\, 5,\, 1,\, 3)$$

This is useful in tour problems where there is no specified start and end positions and the tour is closed by connecting the last vertex to the first.

By regarding each of the entries in the adjacency matrix as individual Bernoulli trials, the random shuffling process X_{rand} can be characterized by the matrix:

$$M_{\text{rand}} = \begin{pmatrix} 0 & \frac{1}{N-1} & \cdots & \frac{1}{N-1} \\ \frac{1}{N-1} & 0 & \cdots & \frac{1}{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N-1} & \frac{1}{N-1} & \cdots & 0 \end{pmatrix}, \tag{C.2}$$

where N is the length of the genome and the entries denoting the probability of the adjacency matrix A noting a connection at the corresponding position.

The entropy rate of this process is:

$$H_{\text{rand}} = H(M_{\text{rand}}) = \sum_i \sum_j M_{ij} \log M_{ij} \tag{C.3}$$

and with (C.2) inserted this yields

$$\begin{aligned} H_{\text{rand}} &= (N^2 - N)\left(-\frac{1}{N-1}\log\left(\frac{1}{N-1}\right)\right) \\ &= N(N-1)\left(\frac{1}{N-1}\log\left(N-1\right)\right) \\ &= N\log\left(N-1\right)\ . \end{aligned} \tag{C.4}$$

This is the entropy rate of a process producing completely random tours, we can call this totally exploratory as no information is preserved. Conversely, a process that does not change the state should be on the other end of the exploration degree scale. The matrix M for that process would consist of only zeros and ones and the entropy would be zero. Thus, the exploration degree is an absolute scale.

The unit of information depends on the logarithmic base. When using the base 2 logarithm, the units are the well known “bits”. This will be the case in the following examples and results, even though the equations will show a log operator with no base number specified.

We will view the mutation operators as random processes and evaluate the entropy rate of them to characterize them on the exploration degree scale.

2.2 Exchange Mutation

The exchange mutation operator is the simpler of the three operators from an analysis point of view, as only two vertices are permuted per operation.

As an example, regard the canonical genome of length 4

$$(1,\ 2,\ 3,\ 4)$$

with the adjacency matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

It has the possible permutations (with the exchange mutation)

$$(2, 1, 3, 4), (3, 2, 1, 4), (4, 2, 3, 1),$$
$$(1, 3, 2, 4), (1, 4, 3, 2), (1, 2, 4, 3)$$

with the corresponding adjacency matrices

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

This results in the M matrix

$$\begin{pmatrix} 0 & 1/6 & 1/3 & 1/2 \\ 1/2 & 0 & 1/6 & 1/3 \\ 1/3 & 1/2 & 0 & 1/6 \\ 1/6 & 1/3 & 1/2 & 0 \end{pmatrix}$$

Using (C.3) we obtain an entropy rate of 5,84 bit/generation compared to 6,34 for the completely randomized process.

This result can be generalized by noting that each row contains the same set of probabilities. Each row i represent the isolated view of a single edge going from the i^{th} vertex. To compute the probabilities, the permutations are categorized into three categories:

- The vertex is permuted; connecting the edge to a new vertex.
- The vertex that the edge was going to is permuted; also connecting the edge to a new vertex.
- Neither the source or destination vertex are permuted; no new connections are made.

Note that for the exchange mutation, the amount of possible permutations are given by the binomial coefficient $\binom{N}{2}$, where N is the length of the genome. Note also that the first two categories are equivalent, because either the current vertex are moved to a position "in front of" a new vertex or that vertex is moved to a position "after" the current one. Each of these $N-2$ events results in a probability of $^2/\binom{N}{2}$, with the exception of the vertex that was in front the current vertex as this results in a probability of $^3/\binom{N}{2}$. The rest of the permutations leaves the edge intact, resulting in a single entry with the probability of $\binom{N-2}{2}/\binom{N}{2}$ each. As these probabilities are per row, the result must be multiplied by the number of rows. Thus, the formula for computing the entropy rate for the exchange mutation process of a genome with length N becomes:

$$\begin{aligned} H_1 &= -\frac{3}{\binom{N}{2}} \log\left(\frac{3}{\binom{N}{2}}\right) \\ H_2 &= -(N-3)\frac{2}{\binom{N}{2}} \log\left(\frac{2}{\binom{N}{2}}\right) \\ H_3 &= -\frac{\binom{N-2}{2}}{\binom{N}{2}} \log\left(\frac{\binom{N-2}{2}}{\binom{N}{2}}\right) \\ H_{\text{exchange}} &= N\left(H_1 + H_2 + H_3\right) \end{aligned} \tag{C.5}$$

2.3 Inversion Mutation

The exchange and the inversion share some of the same mechanics. Two distinct indices are chosen at random and the contents of the genome at the selected indices are exchanged. However, the inversion mutation also exchanges all content between the indices. This introduces a probability of exactly 0.5 that a given entry in the adjacency matrix will be changed to the index that previously pointed to that entry, which essentially is a transposition of the affected part of the adjacency matrix.

Where the exchange mutation had $\binom{N}{2}$ possible combinations, the inversion mutation has $N^2 - N$ possible combination. This leads to the entropy rate

$$\begin{aligned} H_1 &= -\frac{N^2-5N+8}{2(N^2-N)} \log\left(\frac{N^2-5N+8}{2(N^2-N)}\right) \\ H_2 &= -(N-2)\frac{2}{N^2-N} \log\left(\frac{2}{N^2-N}\right) \\ H_3 &= -\frac{1}{2}\log\left(\frac{1}{2}\right) \\ H_{\text{inversion}} &= N\left(H_1 + H_2 + H_3\right) \end{aligned} \tag{C.6}$$

2.4 Displacement Mutation

The displacement mutation is different from the two other mutations in that it internally uses three random processes; one deciding the start of the subsection to move, one deciding the length of the subsection, and one deciding how far to displace it. There are three ways that an entry in the adjacency matrix may change: Either

- a subsection is moved "in front" of the corresponding index, or
- a subsection starting at the index "in front" of the corresponding index is moved to another place in the genome, or
- the corresponding index is the last index in a subsection being moved.

A visualization of these three cases are shown in Fig. C.1.

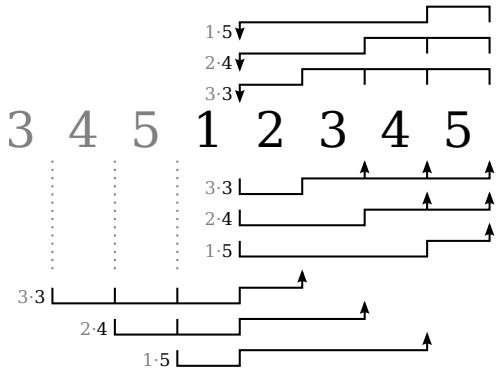


Fig. C.1: The displacement mutation working on a five-entry sequence. The possible mutations changing the adjacency matrix for row 1 is shown. An arrow indicates that a subsection will be inserted here. Next to the subsections, black numbers indicate the new column index that will be set to 1 in the adjacency matrix row 1, and the gray number indicated the number of instances. Some of the sequence is copied to the front of the sequence in order to visualize the mutation "wrapping" around the sequence.

This leads to each probability being a multiple of three over the total possible permutations C. Note that the longer the subsection is, the fewer are the possibilities to insert it back into the genome, thus indices "further away" from an index has less likelihood of becoming the next adjacent index. The total combinations that changes an adjacency entry is K, thus the likelihood of a mutation not changing the entry for

an index is $1 - {}^{K}/_{C}$. The total entropy rate may be calculated by:

$$\begin{aligned} C &= {}^{1}/_{2}N^3 - {}^{3}/_{2}N^2 - N \\ K &= {}^{3}/_{2}N^2 - {}^{9}/_{2}N + 3 \\ H_1 &= -\sum_{n=1}^{N-2} \frac{3n}{C} \log\left(\frac{3n}{C}\right) \\ H_2 &= -\left(1 - {}^{K}/_{C}\right) \log\left(1 - {}^{K}/_{C}\right) \\ H_{\text{displace}} &= N\left(H_1 + H_2\right) \end{aligned} \tag{C.7}$$

2.5 Monte Carlo Simulations

A Monte Carlo simulation is performed to verify the formulas. This is done by applying the mutation operator to an initial genome g_{init} several times and adding the resulting adjacency matrices. By dividing the cumulation matrix by the number of trials, an approximation of the adjacency probability matrix M in (C.2) is achieved, which then can be used to compute the entropy rate. See Algorithm 5.

Algorithm 5 Monte Carlo simulation to determine the entropy rate of mutation operators.

Require: Genome and number of iterations: g_{init}, n
  $M \leftarrow$ ZeroMatrix
  for $i \leftarrow 0, n$ **do**
    $g \leftarrow \text{Mutate}(g_{\text{init}})$
    $A \leftarrow \text{AdjacencyMatrix}(g)$
    $M \leftarrow M + A$
  end for
  $M \leftarrow \frac{M}{n}$
  $H \leftarrow \text{Entropy}(M)$
  return Entropy rate: H

3 Results

A comparison of the entropy rates of the random process in (C.4), the exchange mutation process in (C.5), the inversion mutation process in (C.6), and the displacement mutation process in (C.7) is shown in Figure C.2. The Monte Carlo simulations produce the same results as shown Fig. C.2.

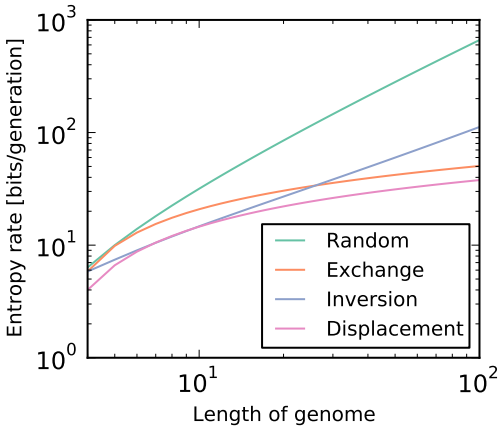


Fig. C.2: Entropy rate of the random, exchange, inversion, and displacement mutation processes as a function of genome length for 4–100. The linear tendency continues for larger lengths of genomes.

4 Discussion

As expected, the random process has the highest entropy rate. Nothing has a higher entropy rate than a completely random process.

The exchange and the displacement operators exhibit the same general behavior in Figure C.2, the exchange mutation having a slightly higher entropy rate. This is not surprising as the exchange mutation "breaks" 4 links when exchanging two entries, i.e. the two entries that are being moved and the two entries that the receives them, where the displacement mutation breaks only 3 links, namely the one where the section is moved from, the one where it is move to, and the last entry of the section.

The inversion mutation has a rather high entropy rate. This is because of the fact that this analysis is based on the asymmetric case, i.e. (1, 2, 3) is not the same as (3, 2, 1), which is the case in the symmetric case. So the inversion mutation breaks all the links in the section being inversed, hence the high entropy rate. This explains the difference in shape from the exchange and displacement mutations in Figure C.2 as the number of entries being affected by the inversion mutation scales with the number of entries in the genome, whereas the other two has a constant number of affected entries.

The result of this analysis may be used to control the diversity of the population in a genetic algorithm like described in [3]. Now, by knowing the entropy rate of the mutations, a control law can be constructed, which determines the needed mutation rate to to control the diversity given a specific mutation.

The three mutations described is not only usable in genetic algorithms, they may be used in any meta-heuristic algorithm using neighborhood functions. Thus, the entropy rates determined here are also usable in those algorithms.

5 Acknowledgments

This work is supported by the Danish Council for Strategic Research under grant no. 09-067027 (ASETA). See `www.aseta.dk` for more details.

References

[1] P. Larrañaga, C. Kuijpers, R. Murga, I. Inza, and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[2] K. Maekawa, N. Mori, H. Tamaki, H. Kita, and Y. Nishikawa, "A genetic solution for the traveling salesman problem by means of a thermodynamical selection rule," in *Proceedings of IEEE International Conference on Evolutionary Computation.* IEEE, 1996, pp. 529–534.

[3] Y. Tsujimura and M. Gen, "Entropy-based genetic algorithm for solving TSP," in *1998 Second International Conference. Knowledge-Based Intelligent Electronic Systems. Proceedings KES'98 (Cat. No.98EX111)*, vol. 2, no. April. IEEE, 1998, pp. 285–290.

Paper D

Adaptive Termination Criterion for Genetic Algorithms Solving Traveling Salesman's Problems

Karl Damkjær Hansen, Anders la Cour-Harbo

The paper is submitted to be included in the
Proceedings of the Genetic and Evolutionary Computation Conference 2014, GECCO'14

Abstract

An adaptive termination criterion for genetic algorithms is presented. This termination criterion is specifically designed for algorithms solving travelling salesman's problems in robotic path planning. This criterion helps roboticists without expert knowledge of genetic algorithms by hiding implementation specific parameters of genetic algorithms; instead it uses a model of the computation time of the algorithm and automatically terminates the computation when a suitable solution has been found. Like traditional termination criteria, this criterion observes the scores of solutions as the algorithm is running. It then fits a decaying function model to the history of the scores and determines the optimal stopping time. The model for the development of this score over time is developed and the criterion is evaluated in the context of path planning for an autonomous helicopter and shows promising behavior.

1 Introduction

This work attends the problem of stopping genetic algorithms at the right time; the stopping criterion. The resulting criterion implicitly takes variables like problem size, processing power, and algorithm implementation into account; things that are difficult for a non-expert to evaluate. All it needs from the user is a cost function on the computation time, which is more intuitive to compute for non-experts.

As the focus is on the stopping criterion, not much regard is given to the specific implementation or performance of the genetic algorithm per se. As a matter of fact, the stopping criterion implicitly takes the performance of the genetic algorithm into account, so even a mediocre implementation of the algorithm will benefit from it.

1.1 Background

The ASETA project [1, 2] will try to accomplish a reduction in pesticide usage in agriculture. The approach is to use unmanned aircrafts (UAs) for surveying the fields and pinpointing weed infestations using image analysis. Unmanned ground vehicles (UGVs) will then go to these spots and cut or spray the individual plants. The rationale for this approach is to reduce the search space for the UGVs to areas with high probability for finding weed infestations so that they will not have to analyze parts of the field without weed infestations. It is in the context of this project that this research has been carried out.

1.2 Traveling Salesman's Problem

A part of the ASETA project is the path planning for the robots. The route for the UAs are composed of waypoints that they visit in sequence. The route for the robot

must be short in order to, not just save time, but also conserve energy. This problem is called the traveling salesman's problem (TSP).

If a traveling salesman is going to visit a collection of cities. He is interested in finding the shortest possible route between them, this will save him time traveling and give him more time to sell his products.

The TSP is easy to understand but hard to solve. In 1972, Richard M. Karp proved 21 combinatorial problems to be NP-complete [3]. Among these problems was the Hamiltonian Cycle, a special case of the TSP. This implies that the TSP belongs to a class that is at least as hard as NP-complete; this class is called NP-hard. This property means that the problem becomes impractical to solve optimally when the number of waypoints increases.

For robotic path planning, the TSP arises in many forms where the robot has several objectives that it must travel between. Some examples are the watchman route problem [4], the shortest safari route problem [5], and the zookeeper route problem [6]. Along with many others, they have the property that the solution to the problem is a traversable route for an agent e.g. a mobile robot.

1.3 Genetic Algorithms

Genetic algorithms can be used to find good solutions to the TSP [7, 8]. These solutions are often not optimal but close to it. Genetic algorithms belong to the family of local search algorithms. They work by starting out with a feasible solution, which is not necessarily very good, and iteratively improve on this solution, often ending up close to the optimal.

Genetic algorithms are based on the concept of Darwinian evolution. While iterating though generations of a pool of solutions, called the genomes, the algorithm selects the most fit solutions and breeds them to produce better solutions. J. H. Holland is often attributed with the definition of the genetic algorithm [9], although current implementations of genetic algorithms do not always strictly follow his approach. The general genetic algorithm has three parts: First, a breeding part; then, a mutation part; and lastly, a survival-of-the-fittest part. These parts will continuously improve on the pool of solutions much like we know it from biology.

Larrañaga et al. [10] describes various operators and representations of genetic algorithms for solving TSPs. A popular representation is the path representation where each genome is simply represented as the sequence of waypoints that must be visited. A number of convenient mutations have been developed for this representation, among these; the displacement, exchange, and inversion mutations [11–13]. These will be the workhorses in the example shown later.

The genetic algorithm is an anytime-algorithm; we can stop it at any time and still get a feasible solution. Usually, the algorithm is stopped when the solution is not improving much per generation. This stopping criterion is set by the designer. If, for

some reason, the system needs a solution before the criterion is reached, it can just stop the algorithm, and because of the anytime behavior it will receive a feasible solution. On the other hand, if we let the genetic algorithm run, it will never stop and in the end it will revisit the solutions it has previously looked at.

The stopping criterion of the genetic algorithm should be tuned to the specific problem. If we let it run for too long, it will waste time and computational power. However, if we stop it too soon, the solution will be bad and the traveling salesman will use too much time traveling. The traditional stopping criteria are the stall-time and stall-generations criteria. These criteria evaluate the development of the solution in either a given time frame or over a given number of generations. When the rate of change in the solution quality falls below a set limit; the algorithm terminates.

The parameters for the stopping criterion should be tuned so that they match the type and size of the problem, the implementation of the genetic algorithm, and also the processing power of the computer. If the problem is large, i.e. the flight time of the resulting TSP solution is long, the computation time might be insignificant and the termination limits on the stall-time or –generation criteria can be set fairly large, conversely for smaller problems. If the mutations of the genetic algorithm are very effective, a good solution is quickly obtained, and the stall-time can be set short. If the processor is slow the stall-time must be set long, as the algorithm needs more time to converge.

1.4 An adaptive stopping criterion

Having expert-knowledge, it should be possible to set the stall-time or –generations parameters satisfactory for a fixed problem type with a fixed type of computer. If the problems presented to the algorithm differs in size and composition, it is difficult to set one specific termination limit at design time; rather, we will let the algorithm itself figure out when it starts to waste time and resources, and terminate automatically.

The stopping criterion presented here seeks to optimize the total time consumption of a system that both computes the path of the traveling salesman and executes it. If a system must both find a solution and execute that solution in the least amount of time, the stopping criterion must not only be conditioned on the quality of the solution alone, but also on the time it takes to come up with that solution. It might be best to stop the algorithm sooner, in order to give the execution part more time to complete a slightly worse solution. The approach described here anticipates how the solution quality improves as the algorithm is running. It then evaluates when it is best to stop computing and start executing. The anticipation process is continuously refined with each generation, and fits easily into any general genetic algorithm.

This termination criterion has the further advantage of hiding implementation specific details from the user of the genetic algorithm. A roboticist who wants to use a genetic algorithm might not have the insight into the inner workings of it. She will then

have a hard time figuring out which parameters to tune in order to have an effective stopping criterion. The presented stopping criterion uses cost functions to describe the computational time and the traveling time, so that no tuning is necessary.

The system that uses this genetic algorithm and stopping criterion consists of one single UA and a base station presented with a collection of waypoints, which the UA must visit. The setup is such that the plan must be calculated first, before the UA launches.

The density of the waypoints must be relatively high. If the waypoints are very far apart or the UA flies very slowly, the time it takes to compute the solution will be insignificant next to the time it takes to execute the resulting plan, and we will gain very little by optimizing the termination. With the current amount of work being done with indoor flights, for example by Vijay Kumar et al. at University of Pennsylvania [14], the high density problem becomes more interesting.

2 Methods

The stopping criterion is designed for genetic algorithms solving TSPs. The genetic algorithm used in this work is using a path representation, the three mutations described above, and Order Crossover. Although there is no proof, we assume that the algorithm converges to a near-optimal solution. Note, that the actual implementation of the genetic algorithm is not under scrutiny here, thus this very simple model is used.

In this section, a model for the evolution of the genetic algorithm is presented. The model is used to predict how the algorithm will perform as it is running. The parameters for this model are estimated along the way using nonlinear regression. The value of the objective function of the genetic algorithm is compared to a simple cost function of the computation time. In the end of this section, two additional possible cost functions are presented.

When an agent is presented with a TSP, the total time consumption for that agent consists of the time spent by the genetic algorithm solving the TSP and the time spent executing the computed solution. This is expressed as

$$T_{\text{tot}} = T_{\text{GA}} + T_{\text{exe}}. \tag{D.1}$$

As time goes and more generations are bred, the quality of the solutions found by a genetic algorithm solving a TSP approaches that of the optimal solution. Here the quality is expressed in time. Later, the quality will be length, as we are looking for the shortest path, however, this will be corrected by dividing with the speed of the vehicle.

Each generation takes a specific time to compute. Consequently, the number of generations computed is linear with time t:

$$T_{\text{GA}} = \epsilon k = t \tag{D.2}$$

where ϵ is the time it takes the computer to evaluate one generation, and k is the number of computed generations.

Next step is to evaluate the execution time T_{exe} as a function of computed generations. When genetic algorithms solve TSPs, the quality gradually get better. Fig. D.1 shows the result of 1000 genetic algorithms solving the Berlin 52 problem from the TSPLIB package [15], this shows a rapid convergence in the beginning slowing down towards the end. To model this behavior, we will look at the solution space of the TSP.

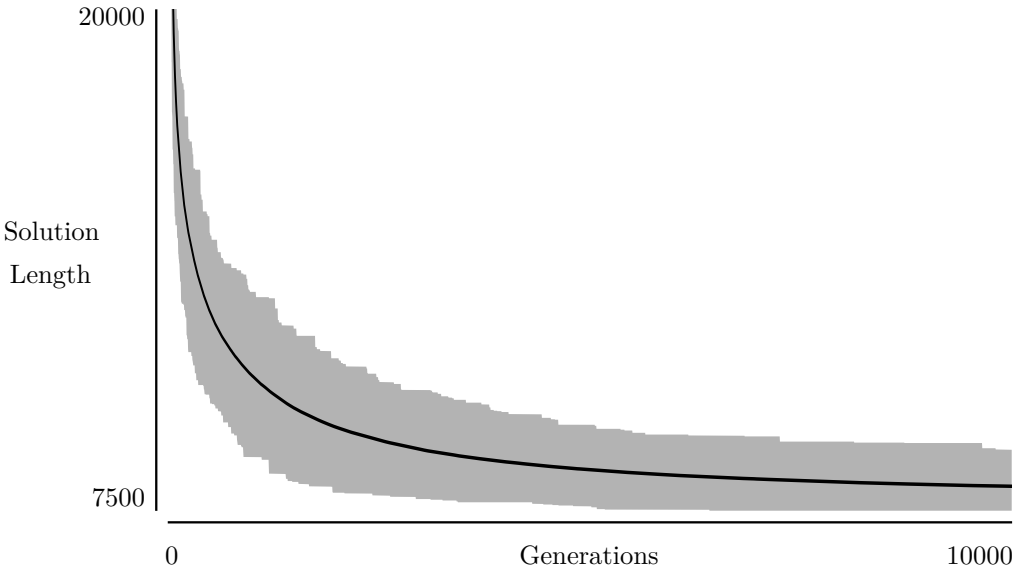


Fig. D.1: Solution quality in length vs. generations for 1000 genetic algorithms solving the Berlin 52 problem from the TSPLIB package [15]. Maximum, minimum, and average values for each generation is shown.

2.1 Model

genetic algorithms can be used to solve many different optimization problems, some of which can have very "hostile" solution spaces where a single mutation of a genome may improve the solution quality significantly. Think of a flat landscape with a number of holes in it, one needs to be very (un-)lucky to fall into one of the holes. Luckily the TSP generally seems to have a nice curvy landscape that may guide the genetic algorithms along a gradient. In Fig. D.2 a histogram of the solution quality for all the solutions to a 10-city problem are shown. Naturally this is a generalization, as the problem is 10-dimensional and the histogram is 1-dimensional. However, it may give an intuition about the type of landscape the problem presents.

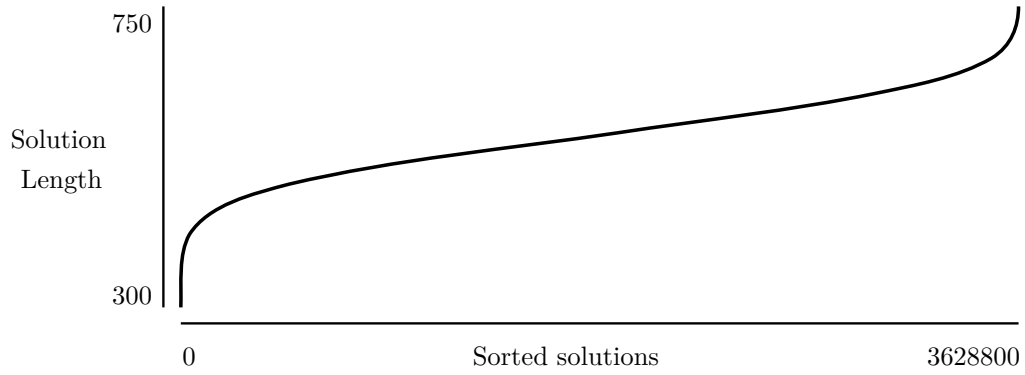


Fig. D.2: Histogram of the complete enumeration of all solutions to a 10 city TSP sorted according to quality. The cities were uniformly randomly distributed over a 100 x 100 area and the quality values are the Euclidean distances of the complete tour.

Let's brutally model this histogram as a straight line segment starting at $(0, s^*)$ ending in $(N, s^* + \alpha N)$, where s^* is the length of the shortest solution 0, and α is the slope that makes $s^* + \alpha N$ the length of the longest solution N. Then for any solution, a number of possible mutation operators can be applied. These will either shorten of lengthen the solution length. The mutations used will define an interval on the x axis, as they will have a maximum to the increase or decrease they are able to perform on the given solution. This interval is called the neighborhood of the current solution. Different mutations will create different neighborhoods. A uniform distribution is assumed within the neighborhood, which gives rise to the model seen in Fig. D.3.

Region 1 of the model is the most interesting of the three, as this is the area where the genetic algorithm converges to the optimal solution. A zoom-in on regon 1 is shown in Fig. D.4. Here the current solution n_k of the k^{th} generation has the length s_k and the expected value of n_{k+1} can be evaluated to the grayed out area divided by the number of solutions spanning that area, N_m. Δ_m is the difference between the longest and the shortest mutation. The genetic algorithm will favor better solutions, so the model rejects any solutions that are longer than the current. So the gray area can be evaluated as a rectangle less a triangle.

$$\mathrm{E}\left[s_{k+1}\right] = \frac{N_m s_k - \frac{n_k \alpha}{2} n_k}{N_m} = s_k - \frac{\alpha n_k^2}{2N_m} \tag{D.3}$$

$$\alpha n_k = s_k - s^* \tag{D.4}$$

$$\Delta_m = \alpha N_m \tag{D.5}$$

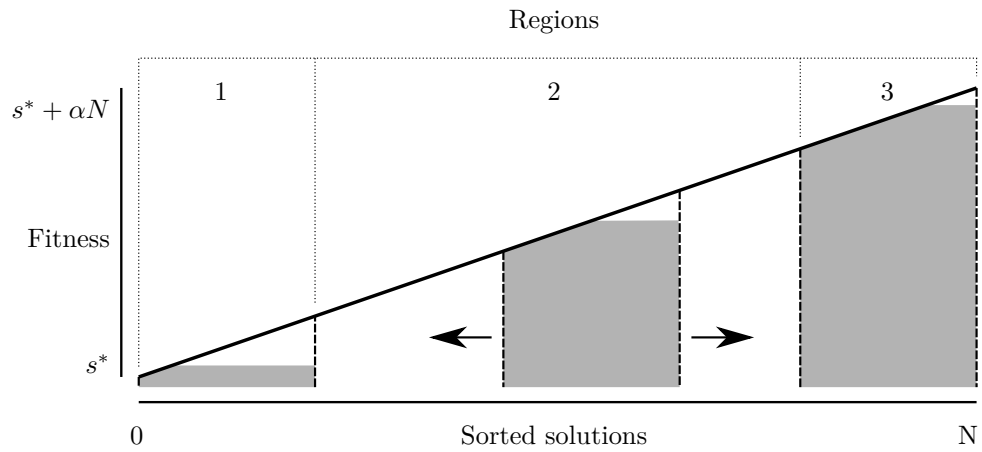


Fig. D.3: Model of the histogram seen in Fig. D.2. Mutations may move a solution in its neighborhood. Three regions are defined, region 1, where the solution cannot get better than s^*, region 2 where there is a constant improvement, and region 3, which is the reciprocal of region 1.

Equations (D.3), (D.4), and (D.5) yield,

$$\mathrm{E}\left[s_{k+1}\right] = s_k - \frac{\left(s_k - s^*\right)^2}{2\Delta_m} \tag{D.6}$$

which is transformed to the difference equation,

$$\frac{\Delta \mathrm{E}\left[s\right]}{\Delta k} = -\frac{\left(s_k - s^*\right)^2}{2\Delta_m} \tag{D.7}$$

Now, let's consider a differential equation,

$$\frac{ds}{dk} = f\left(k\right) g\left(s\right) \tag{D.8}$$

where:

$$f\left(k\right) = -\frac{1}{2\Delta_m},$$
$$g\left(s\right) = \left(s - s^*\right)$$

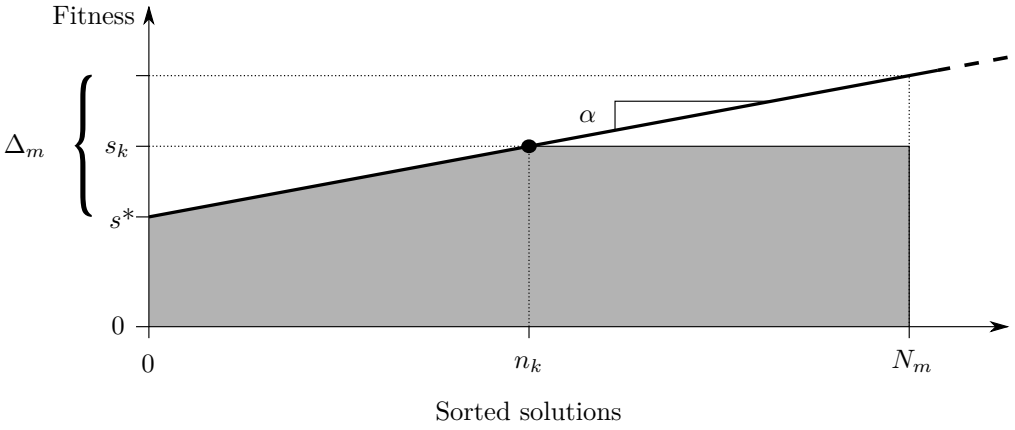


Fig. D.4: Model of the lower part of the sorted solutions, where a mutation may move a given solution at generation k, n_k, to a solution in the interval $[0; N_m]$. A resulting solution that is worse is rejected, thus capping the possible quality value at s_k.

The solution to this equation is obtained by,

$$\begin{aligned}
\int \frac{1}{g(s)} ds &= \int f(k)\, dk \\
\int \frac{1}{(s-s^*)^2} ds &= \int -\frac{1}{2\Delta_m} dk \\
\int \frac{1}{(s-s^*)^2} d(s-s^*) &= -\frac{k}{2\Delta_m} + C_k \\
-\frac{1}{(s-s^*)} + C_s &= -\frac{k}{2\Delta_m} + C_k \\
s &= \frac{2\Delta_m}{k - 2\Delta_m (C_k + C_s)} + s^*
\end{aligned} \tag{D.9}$$

The length of the solutions to the TSPs, s, are represented in distance. The solution is divided by the speed of the vehicle to get the execution time:

$$T_{\text{exe}} = \frac{s}{v} \tag{D.10}$$

Minimizing the total time consumption yields the optimal stopping generation k_{stop} for the genetic algorithm. The combination of (D.1), (D.2), and (D.9) with (D.10) yields:

$$T_{\text{tot}} = \frac{2\Delta_m}{v\,(k - 2\Delta_m (C_k + C_s))} + \frac{s^*}{v} + \epsilon k, \tag{D.11}$$

which has a strictly monotonic increasing derivative,

$$\frac{d}{dk}T_{\text{tot}} = -\frac{2\Delta_m}{v\left(k - 2\Delta_m\left(C_k + C_s\right)\right)^2} + \epsilon, \tag{D.12}$$

Solving for 0 yields:

$$k_{\text{stop}} = \sqrt{\frac{2\Delta_m}{v\epsilon}} + 2\Delta_m\left(C_k + C_s\right) \tag{D.13}$$

Now, by finding Δ_m, v, ϵ, C_k, and C_s it is possible to calculate the best k_{stop}. The values are estimated by fitting the expression in (D.9) to the solution quality reported by the genetic algorithm as it is running. Doing so will result in increasingly better estimates of the values and, by using (D.13), the stopping generation will converge to the actual k_{stop}. It is important to note, however, that the actual evolution of the solution quality only approximately follows (D.9), any mismatch between the actual solution and the model will give an erroneous k_{stop}.

The parameters v and ϵ are constants whereas Δ_m, C_k, and C_s are estimated. As C_k and C_s are integration constants they can be lumped together as C. This yields an expression for k_{stop} as a function of the current generation, in other words a prediction of the stopping time,

$$k_{\text{stop}}\left(k\right) = \sqrt{\frac{2\Delta_m\left(k\right)}{v\epsilon}} + 2\Delta_m\left(k\right)C\left(k\right) \tag{D.14}$$

This is visualized for one of the instances from Fig. D.1 in Fig. D.5. Where the termination error Δ_k is shown. In this example the algorithm terminates a bit late; it should have stopped at $k = 5867$ but stopped at 6236.

2.2 Cost Functions

The requirement of the stopping criterion is that the designer presents it with two cost functions. One is the objective function of the genetic algorithm; the other is the cost of the computational time. In the calculations above it was assumed that the cost of one second of computational time, was equivalent to one second of execution time. Therefore $t = \epsilon k$ was simply added to (D.11) to yield (D.13). It is important that the two functions use the same unit of measure. In the description of the criterion above, the common cost was time.

The problem of attributing cost to time has been treated in both economics and behavioral sciences. Even though the opinions differ on how humans understand time, it seems that there is a consensus that people tend to get more frustrated when subjected to long waits [16] and prefer advancing the timing of satisfaction rather than postponing it [17]. This seems to undermine the notion that the cost of running a genetic algorithm for 10 minutes is only twice as expensive as running it for 5 minutes.

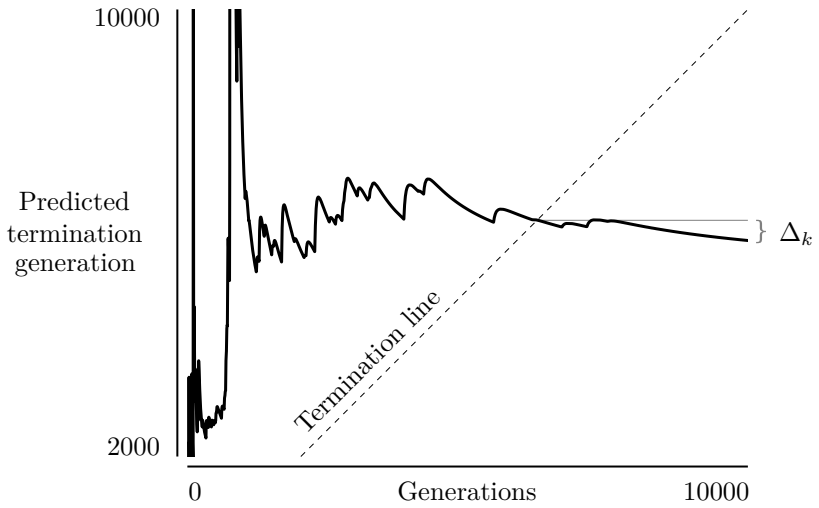


Fig. D.5: The predicted termination generation as a function of computed generations. When the algorithm crosses the termination line, the current generation count surpasses the termination generation and the algorithm should stop. Here it continues for a while to show the termination error Δ_k.

Time is also not the best measure for the objective function of the genetic algorithm. In the case of routing for a flying robot, it will not be equally expensive to fly 10 meters vertically up and 10 meters horizontally, even though it may take the same time. The measure might rather be a combination of energy usage, wear and tear, as well as time.

2.3 Other cost functions for computation time

The nature of the genetic algorithm is to provide better and better solutions. So the cost function of the algorithm will always be monotonic decreasing. To guarantee a stopping time the cost function on computation time must be increasing. In (D.2) the incline was set to 1, this is easily extended with any linear cost function $\text{cost}(t) = \xi t = \xi\epsilon k$:

$$k_{\text{stop}} = \sqrt{\frac{2\Delta_m v}{\epsilon\xi}} + 2\Delta_m (C_k + C_s) \tag{D.15}$$

This can be used to model computational time having a different cost than execution time.

3 Discussion

The presented termination criterion works with cost functions, which relates directly to real life measures, such as salary, cost of energy or wear and tear. With the traditional stopping criteria, such as stall-time and stall-generations, the designer has to decide on a limit on the rate of change in the solution and either a time frame or a number of generations to evaluate over. These parameters, especially the rate of change, are not immediately possible to decide on, if the designer does not know about the mechanics of genetic algorithms. We think that the cost function is easier to evaluate for the designer.

The termination criterion is more robust to changes in the executing platform. Both of the traditional stopping criteria are sensitive to changes in the computation time. The new criterion is robust to these changes. Consequently, the criterion is portable and can be run on both fast and slow computers without any fine-tuning of the parameters.

The main drawback of the new criterion is that it might stop at local minima, and start the execution too early. Even though the model is convex and only has one minimum, the model parameters are estimated from the output of the genetic algorithm, which is not strictly decreasing. This, however, is also the case in the stall-generations and -time criteria.

In order to fit the nonlinear model in (D.9), nonlinear regression is used. In this way, several iterations of the fitting procedure may be needed to find a good approximation. It is a concern that this process will consume much computing power compared to the genetic algorithm.

4 Conclusion

A termination criterion for genetic algorithms solving travelling salesman's problems in a robotic setting has been presented. Tests have shown the criterion to terminate the genetic algorithm such that the found solution is within 5 % of optimal. This is a positive result as this enables a roboticist without expert knowledge of genetic algorithms to implement these and have confidence in achieving satisfactory results. Expert users also have an advantage of using this termination criterion as it adapts to different sizes and types of TSPs, which is convenient in an autonomous system working in an unknown environment.

5 Acknowledgments

This work is supported by the Danish Council for Strategic Research under grant no. 09-067027 (ASETA). See `www.aseta.dk` for more details.

References

[1] A. la Cour-Harbo, "ASETA Project," 2010. [Online]. Available: http://www.aseta.dk

[2] W. Kazmi, M. Bisgaard, K. D. Hansen, F. Garcia-Ruiz, and A. la Cour-Harbo, "Adaptive Surveying and Early Treatment of Crops with a Team of Autonomous Vehicles," in *Proceedings of the 5th European Conference on Mobile Robotics*, 2011, pp. 253–258.

[3] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum Press, 1972, pp. 85–103.

[4] W. Chin and S. Ntafos, "Optimum watchman routes," in *Proceedings of the second annual symposium on Computational geometry - SCG '86*. New York, New York, USA: ACM Press, 1986, pp. 24–33.

[5] X. Tan and T. Hirata, "Finding shortest safari routes in simple polygons," *Information Processing Letters*, vol. 87, no. 4, pp. 179–186, Aug. 2003.

[6] C. Wei-Pang and S. Ntafos, "The zookeeper route problem," *Information Sciences*, vol. 63, no. 3, pp. 245–259, Sep. 1992.

[7] D. Johnson and L. McGeoch, "The traveling salesman problem: A case study in local optimization," in *Local search in combinatorial optimization*. Wiley Publishing, Inc., 1997, pp. 215–310.

[8] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, Jun. 2006.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975, vol. Ann Arbor, no. 53.

[10] P. Larrañaga, C. Kuijpers, R. Murga, I. Inza, and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[11] D. B. Fogel, "Applying Evolutionary Programming to Selected Traveling Salesman Problems," *Cybernetics and Systems*, vol. 24, no. 1, pp. 27–36, Jan. 1993.

[12] W. Banzhaf, "The "molecular" traveling salesman," *Biological Cybernetics*, vol. 64, no. 1, pp. 7–14, Nov. 1990.

[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin-Heidelberg: Springer-Verlag, Apr. 1999, vol. 4, no. 2.

[14] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP Multiple Micro-UAV Testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, Sep. 2010.

[15] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, Nov. 1991.

[16] J. A. Jacko, A. Sears, and M. S. Borella, "The effect of network delay and media on user perceptions of web resources," *Behaviour & Information Technology*, vol. 19, no. 6, pp. 427–439, Jan. 2000.

[17] T. Koopmans, "Stationary ordinal utility and impatience," *Econometrica: Journal of the Econometric Society*, vol. 28, no. 2, pp. 287–309, 1960.

Paper E

Aircraft Mission Planning with Refueling Constraints using Genetic Algorithms

Karl Damkjær Hansen, Anders la Cour-Harbo

The paper is submitted to be included in the
Proceedings of IEEE Conference on Intelligent Systems 2014

Abstract

Fuel is a limited resource. When planning long missions for unmanned aircraft systems, refueling stops must be considered. An otherwise distance-optimal path might be infeasible without refueling, but the added fuel consumption of going to a refueling station and resuming the work renders the optimal path sub-optimal. This work describes an approach to path planning for traveling salesman's problems using waypoints including refueling stops at specified locations. The planning is done using a genetic algorithm that optimizes the length of the path. Refueling stops are randomly inserted into or removed from the solutions in the population. Otherwise, the algorithm treats the depots as regular waypoints. Some measure of infeasibility is allowed in the population, but the final product of the planning is a feasible plan for the aircraft.

1 Introduction

The trend in robotics is to work autonomously for longer and longer periods of time. However, when the durations of the missions of the robots are so long that the fuel may run out, refueling must be considered in the planning of the mission. An example of this long-term autonomy is persistent surveillance [1], where a group of unmanned aircraft (UA) takes turns to keep observation on a specified surveillance location. When one UA is observing, the others are either in transport to or from the base or recharging.

Many robotic applications require human intervention when the robot needs to be refueled or recharged. For some applications it is not practical or maybe even impossible to have human interventions during the mission, for others it is reasonable enough for the robot to simply present itself at a refueling station, where a human operator will manage the refueling process. For autonomous refueling of electrically driven UAs, two roads have been investigated. One is recharging the on-board batteries [2], the other is to swap the batteries [3].

The fuel constrained traveling salesman problem studied here, is interesting in robotic long-time autonomy applications. This problem is related to the persistent surveillance problem in [1], however here the UA must survey a piece of land by taking aerial images at several locations to get a complete coverage of it. Sundar and Rathinam have looked at a similar problem [4]. They have named it Fuel Constrained, UAV Routing Problem (FCURP). The principal interest of Sundar and Rathinam, however, was to route UA with small fuel capacities between target waypoints further apart than the UA could fly on one tankful of fuel by "hopping" from one refueling depot to the next. In the surveying problem at hand, the targets are many and densely distributed such that the UA can not visit all of them on one tankful and will have to go back to the refueling depot. But despite the difference in the problems, the problem definition in [4] is much the same.

1.1 Problem Definition

The problem definition is almost like that of Sundar and Rathinam [4], with the exception of a fixed start and stop location. Consider a set of target waypoints T and a set of depots D. Let $V = T \cup D$ be the vertices of the fully connected graph $G = (V, E)$. The fuel consumption for moving between vertex $i \in V$ to $j \in V$ is denoted f_{ij}. The fuel consumption is assumed to satisfy the triangle inequality. L denotes the maximal fuel capacity of the UAS.

The objective is to find a tour TOUR $:= (v_1, v_2, \cdots, v_p)$ in G that visits all T once and each D zero or more times while obeying the fuel capacity L and minimizing fuel usage. Formally:

- TOUR $\supseteq T$
- For any subsequence of TOUR starting at a depot d_1 and ending at the next depot in line d_2, $(d_1, t_1, t_2, \cdots, t_p, d_2)$, the fuel usage must be at most equal to L, i.e. $f_{d_1 t_1} + \sum_{i=1}^{k-1} f_{t_i t_{i+1}} + f_{t_k d_2} \leq L$
- Minimize the total fuel usage, $\sum_{i=1}^{p-1} f_{v_i v_{i+1}} + f_{v_p v_1}$

In the following, a method to solve this problem is proposed. It is based on a genetic algorithm for solving traveling salesman problems (TSPs), the main contribution is a set of mutations to handle refueling stops, a cost function that computes fuel usage, and a termination criterion that ensures a feasible solution.

2 Methods

This method is based on a genetic algorithm (GA) approach. See [5] for a comprehensive survey of the possible GA operators in TSP. Here, the path representation is used along with the classical displacement and inversion mutation operators [5]. Other than these mutations, two extra types of mutations are introduced to manage the insertion and extraction of refueling stops. An example of the genetic algorithm used is outlined in Fig. E.1.

2.1 Path Representation

The genomes are simply represented as a sequence of waypoints

$$(v_1, v_2, \cdots, v_p).$$

This translates to a route going from v_1 over v_2 to v_p. Each v_i may be a target or a depot. Note that the traditional path representation does not care about the starting point of the solution as the solution to a TSP is a cycle, so the point following v_p

```
Require: N_p, P_inv, P_dis, P_inj, P_ext
  procedure GENETIC ALGORITHM
    Initialize population ← N_p random genomes
    doTerminate ← False
    while not doTerminate do
      newPopulation ← empty population
      Make ROULETTEWHEEL selector from population
      for i ← 1, N_p do
        child ← ROULETTEWHEEL
        if RAND(0, 1) ≤ P_inv then
          child ← INVERSE(child)
        end if
        if RAND(0, 1) ≤ P_dis then
          child ← DISPLACE(child)
        end if
        if RAND(0, 1) ≤ P_inj then
          child ← INJECT(child)
        end if
        if RAND(0, 1) ≤ P_ext then
          child ← EXTRACT(child)
        end if
        Add child to newPopulation
      end for
      Evaluate newPopulation
      if bestNewGenome > allTimeBest then
        allTimeBest ← bestNewGenome
      else
        worstNewGenome ← allTimeBest
      end if
      population ← newPopulation
      doTerminate ← evaluate termination criterion
    end while
  end procedure
```

Fig. E.1: The genetic algorithm which shows the probabilistic selection of mutation operators and, in the end, an elitism step.

is implicitly v_1. When taking the fuel consumption into account, it seems natural to define a start point where the aircraft if fully refueled as is the case in [4], but this representation does not pose that constraint. This results in a solution that produces a cycle including refueling which the aircraft can repeat continuously. A way to think of it may be that the solution is a steady-state refueling plan. This is reminiscent of a surveillance problem, where the aircraft is patrolling the set of waypoints.

2.2 Displace Mutation

The displace mutation is a random process that picks a section of the genome out and replaces it in another location. E.g. the genome

$$(v_1, v_2, v_3, v_4, v_5, v_6, v_7)$$

would become

$$(v_1, v_4, v_5, v_6, v_2, v_3, v_7)$$

if the mutation replaced the section (v_4, v_5, v_6) after v_1.

Because of the cyclic nature of the problem, the section (v_7, v_1) can also be replaced. If that section is placed after v_3, the result will be

$$(v_2, v_3, v_7, v_1, v_4, v_5, v_6),$$

which is actually the same as above. This leads to an easy implementation, where three different ordered indices are chosen and the section between the first and second index is swapped with the section between the second and third index.

2.3 Inverse Mutation

Like the displace mutation, the inverse mutation also chooses a section of the genome. But instead of moving it around, it reverses the sequence of the entries in it. E.g. the genome

$$(v_1, v_2, v_3, v_4, v_5, v_6, v_7)$$

becomes

$$(v_1, v_2, v_6, v_5, v_4, v_3, v_7)$$

if the mutation inverted the section (v_3, v_4, v_5, v_6). Note that this also equates to inverting the section (v_7, v_1, v_2), as the direction of travel is irrelevant.

2.4 Refueling Mutations

Two refueling mutations are introduced. The first randomly selects a depot from the genome and removes it, the other inserts a random depot at a random position. We give the mutations the names *inject* and *extract* to emphasize that they are more than simple insert and remove operators.

If v_2 is a depot, the inject mutation can transform the genome

$$(v_1, v_2, v_3, v_4, v_5, v_6, v_7)$$

into

$$(v_1, v_2, v_6, v_5, v_4, v_3, v_2, v_7).$$

The extract mutation must not remove the depot entry of a genome if there is only one depot left.

2.5 Mutation Selection

All the mutations are introduced into the GA via a probabilistic selection of mutations. When the GA has selected a genome for mutation a second selection decides which mutation to perform. This is seen in Fig. E.1, where the probability of performing the mutation is determined by P_{dis}, P_{inv}, P_{inj}, and P_{ext}.

2.6 Order Crossover

Genetic algorithms often employ crossover operators to combine parts of two genomes into an offspring genome. A possible crossover for this formulation is the order crossover, which was proposed by Davis [6] for use in epistatic problems. It focuses on preserving the order of the vertices rather than their absolute position in the solution. This also pertains to the representation at hand, as a "good" sequence of vertices are equally good no mater whether it is executed early or late in the solution.

The process is to choose a section at random from the first genome, this is then inserted at the same position in the offspring, then the vertices of the other genome is inserted into the offspring in the same order as the original but without repeating the vertices already inserted. For example, if the genomes

$$(v_1, v_2, v_3, v_4, v_5, v_6, v_7)$$

and

$$(v_4, v_6, v_2, v_7, v_1, v_3, v_5)$$

are combined, choosing the subsection (v_3, v_4, v_5) from the first genome, they will produce the offspring

$$(v_6, v_2, v_3, v_4, v_5, v_7, v_1)$$

As the solution is allowed to visit depots several times, the crossover operation can be allowed to repeat depots when inserting from the second genome, so that if the vertice v_4 is a depot, then the resulting offspring of the above parents will become

$$(v_4, v_6, v_3, v_4, v_5, v_2, v_7, v_1)$$

This refueling aware order crossover may introduce extra refueling stops into the offspring. Even though this may be a step in the right direction towards the best solution, it might not and the extract depot mutation is then needed to counter the bias towards adding depots. Conversely, the crossover could be disallowed to repeat depots, but then the bias would be toward removing depots and the inject mutation is needed to counter the bias.

These counteractions are achieved by increasing the corresponding selection probability P_{ext} or P_{inj}. The best values for these values are not obvious, so the crossover operator is not used in this work, but can be included with some tweaking of the parameters.

2.7 Cost Function and Feasibility

The cost of a genome is evaluated as the total fuel usage of the entire solution. This is simply implemented as the Euclidean distance times a fuel usage constant.

$$f_{v_1 v_2} = c \cdot d(v_1, v_2) \tag{E.1}$$

where d is the Euclidean distance function. The cost of the i^{th} genome is then

$$C_i = f_{sv_1} + \sum_{i=1}^{p-1} f_{v_i v_{i+1}} + f_{v_p s} \ . \tag{E.2}$$

An additional task of the cost function is to evaluate the feasibility of the solutions in terms of satisfying the fuel capacity constraint. This is done by accumulating the fuel usages for each edge between two depot vertices in the solution. E.g. for the subsection

$$(d_1, t_1, t_2, \cdots, t_p, d_2)$$

the fuel usage is

$$F_{d_1} = f_{d_1 t_1} + \sum_{i=1}^{p-1} f_{t_i t_{i+1}} + f_{t_p d_2} \tag{E.3}$$

The accumulated fuel amount is then noted as the amount that should be refueled at the first depot d_1 in the subsection. The procedure is executed for each depot-to-depot

subsection in the solution. If any of the computed refueling amounts are greater than the fuel capacity L, the solution is infeasible.

The implementation of the genomes includes the fuel that is left in the tank at each waypoint, so that the genome

$$((d_1, 8), (t_1, 6), (t_2, 4), (t_3, 2))$$

represents a solution to the three-waypoints one-depot problem with a distance of 2 between each of the points in the sequence.

2.8 Dealing with Infeasible Solutions

This formulation of the algorithm allows infeasible solutions. This is necessary to facilitate an effective exploration of the solution space. I.e. it is not possible to guarantee that the given operators are able to transform any feasible solution to any other feasible solution in a finite number of operations with every intermediate step being feasible. However, if the intermediate steps are allowed to be infeasible by breaking the fuel capacity limit, the displace and inversion mutation can reach all solutions when dealing with problems without depots [7]. Combined with the two refueling mutations, the entire solution space may be explored.

An infeasible solution, however, is not useful as an end product. Usually, a genetic algorithm keeps track of the best solution, but in this case it also needs to keep track of the best feasible solution so that it can be presented to the user when the algorithm terminates.

One serious problem with allowing infeasible solutions is that they will alway produce a lower cost compared to feasible solutions with the same ordering of the target waypoints but depots inserted to achieve feasibility. This means that there is no selection pressure to move towards feasible solutions. This is remedied by imposing a penalty on the infeasible solutions. The penalty is measured as the ratio α of the most severe fuel limit violation compared to the limit L.

$$\alpha = \frac{\max_i F_{d_i}}{L} \tag{E.4}$$

The cost of the infeasible solution is multiplied with the penalty. The feasible solutions, however, are not favored by multiplying penalties of less than 1, as there is no difference between flying with a full tank and a less than full tank.

2.9 Termination Criterion

A much used termination criterion is called the stall-generations criterion, which terminates the algorithm when the best solution has not changed significantly in a preset number of generations. In this case, with presence of infeasible solutions, the termination criterion evaluates only the feasible solutions. This is in order to not terminate the

algorithm when it has found the best infeasible solution as this is not of interest to the user.

3 Results

The algorithm is tested on a problem of waypoints in a circular arrangement of radius 400 with refueling depots located in the center at $[\pm 100, \pm 100]$. The fuel limit was set to 1200. This setup is chosen to visualize the output of the algorithm rather than an emulation of a real life situation, the result is seen in Fig. E.2.

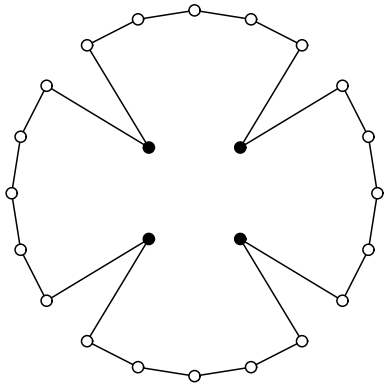


Fig. E.2: The test setup for the algorithm.

For a more real life test case, the berlin52 problem from TSPLIB [8] is used with two refueling depots located at coordinates $[500, 500]$ and $[1000, 500]$. The solution for a fuel capacity of 3000 units per tank is shown in Fig. E.3.

The results were obtained with P_{dis} and P_{inv} set to 0.3, P_{inj} and P_{ext} set to 0.1, and a population of 100 genomes.

The circular solution was found in 3,000 generations, which on an Intel i7 2.8 GHz core took 5 seconds. The berlin52 solution was found in 20,000 generations and took 50 seconds. Depending on implementation, this might be significantly improved.

4 Discussion

This genetic algorithm produces a feasible path for a patrolling-refueling problem by constructing a path that is flyable in a continuous manner.

Practically, the paths can be used by starting the mission from any of the waypoints included in the solution by making sure that the amount of fuel in the aircraft is at least the minimum fuel left figure computed in the cost function.

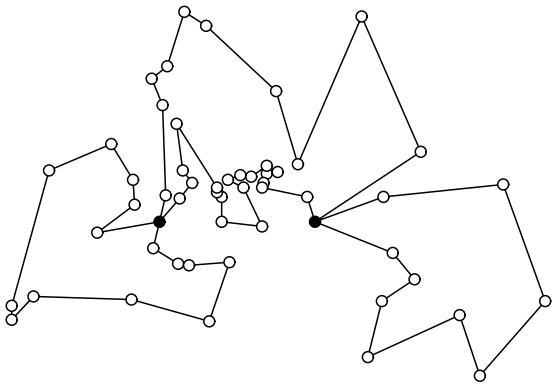


Fig. E.3: The solution to the berlin52 problem from TSPLIB.

If the problem is to be constrained to a fixed initial and finishing waypoint, the mutations can be modified to simply not include the first and last waypoint. In that case, the initial waypoint must be a depot, so that the aircraft can be fueled to the minimum computed amount.

The refueling mutations developed for this problem are non-biased in that they insert and remove depots from the genomes in a uniformly random manner. Heuristic approaches may be evaluated, such as inserting depots only into long sections.

4.1 Future Work

The outline of a modified crossover operator was proposed, but it was not evaluated because of its bias towards generating too few or too many depots. As such, no unbiased crossover operator for this patrolling-refueling problem is known.

Acknowledgments

This work is supported by the Danish Council for Strategic Research under grant no. 09-067027 (ASETA). See `www.aseta.dk` for more details.

References

[1] B. Bethke, J. P. How, and J. Vian, "Group health management of UAV teams with applications to persistent surveillance," in *2008 American Control Conference*. IEEE, Jun. 2008, pp. 3145–3150.

[2] R. D'Andrea, "Flying Machine Arena - Infrastructure," http://www.idsc.ethz.ch/Research_DAndrea/Flying_Machine_Arena/infrastructure, 2014. [Online]. Available: http://www.idsc.ethz.ch/Research_DAndrea/Flying_Machine_Arena/infrastructure

[3] K. A. Swieringa, C. B. Hanson, J. R. Richardson, J. D. White, Z. Hasan, E. Qian, and A. Girard, "Autonomous battery swapping system for small-scale helicopters," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, May 2010, pp. 3335–3340.

[4] K. Sundar and S. Rathinam, "Algorithms for Routing an Unmanned Aerial Vehicle in the Presence of Refueling Depots," *IEEE Transactions on Automation Science and Engineering*, vol. PP, no. 99, pp. 1–8, 2013.

[5] P. Larrañaga, C. M. Kuijpers, R. H. Murga, I. n. Inza, and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[6] L. Davis, "Applying adaptive algorithms to epistatic domains," in *IJCAI'85 Proceedings of the 9th international joint conference on Artificial intelligence*, vol. 1. Morgan Kaufmann Publishers Inc., Aug. 1985, pp. 162–164.

[7] K. D. Hansen, "Characterization of Genetic Algorithm Mutation Operators for Solving Traveling Salesman's Problem," in *Submitted*, 2014.

[8] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, Nov. 1991.

Paper F

Waypoint planning with Dubins Curves using Genetic Algorithms

Karl Damkjær Hansen, Anders la Cour-Harbo

The paper is submitted for publication in the
IEEE Transactions on Automatic Control

Abstract

Mission planning for aircraft is often done as waypoint planning. A sequence of waypoints describing the three-dimensional positions that the aircraft must visit. A common approach is to plan the sequence of the waypoints such that the Euclidean distance between them is minimized. When the high-level waypoint planning is finished, a finer grained planning is executed to obtain a trajectory that the aircraft must follow. When the waypoints in a plan are distributed far apart compared to the turning radius of the aircraft, the two-step planning approach works well, but when the waypoints are closer, the kinematics of the aircraft ruins the plan. This work describes an approach that uses a genetic algorithm to solve the waypoint planning problem while considering the kinematics of the aircraft in one single step. This approach entails the addition of a heading and target speed along with the position in the waypoint definition. The kinematics of the aircraft is modeled with Dubins curves, which are extended to allow variable turning radii.

1 Introduction

When planning a route through a set of waypoints for a mobile robot, the optimization criterion is often the traveled length or time spent traveling. A straight forward method of performing this planning is to optimize Euclidean distance of the route. However, following straight lines between waypoint might easily violate kinematic or dynamic constraints, in fact such a path might be very far from a feasible route for the mobile robot. The inertia of a moving robot implies that some sort of 'turns' might be quite useful for planning a route.

The geometrically simplest of turns is (part of) a circle, which when connected with straight line forms the basis for Dubins curves [1–3]. The Dubins curve is a path composed of line and circle segments, connecting two points with corresponding headings in the plane. Dubins [1] showed that any two points with arbitrary headings can be connected by a combination of two circle segment with a straight line segment in between (CSC), three circle segments (CCC), or any subset of these, with the circle segments C being either left- or right-turning (L or R). The circles in the paths are all constrained to one given radius.

The Dubins curves have also been considered for applications to aircraft [3, 4]. Here the curves to some extent fit the kinematics of an aircraft moving at a constant speed. This is because of the bounded turning speed of an aircraft, where a constant forward speed translates to a constant turning radius. The traditional Dubins curves, defined in the plane, have been generalized to three dimensions [5], where the circle segments no longer are defined as left and right turns, but defined on a doughnut shaped manifold located around the orientation vector of the waypoints. That approach may lead to very

steep inclines, a problem that has been treated in [4] where a bound is set on the rate of change in altitude, leading to 'corkscrew' maneuvers to increase or decrease altitude. A nice review of methods for path and trajectory following of Dubins curves in aircraft autopilots is presented in [6].

The traditional way of planning a waypoint-based path with Euclidean performance function is solving a traveling salesman's problem (TSP), i.e. finding an ordering of the waypoints that produces the shortest possible accumulated distance. The TSP is an $\mathcal{NP}$-hard problem [7] thus exact algorithms are not efficient and heuristics have been developed [8, 9]. Typical approaches include neighborhood-search meta-heuristics like ant-colony optimization, tabu-search, and genetic algorithms (GA).

It seems obvious to combine Dubins curves with a TSP where the performance function is traveled distance along the curves (rather than Euclidean distance). However, the Dubins TSP introduces the continuous variable of heading for each waypoint, and these variables have to be determined somehow in order to determine the Dubins curves connecting the waypoints. Since the length of the path is very much influenced by the headings these variables are part of the optimization. Work has been done to solve this problem by discretizing the headings [3] thus producing a generalized TSP with clusters of waypoints with fixed headings where only one of the waypoints in the cluster should be visited. A well-known heuristic method is known as the alternating algorithm [2], which sets every second heading to point in the direction of the next waypoint in the sequence and the next heading to the same value. This produces a route of alternating straight segments and Dubins curved segments. Another approach is to manage the headings in a genetic algorithm by randomly selecting headings in the neighborhood function [10].

It should be noted that the optimal solution for a Dubins TSP tends to the optimal solution for the Euclidean TSP as turning radius of the aircraft becomes smaller in relation to distance between waypoints. This is because the turning then becomes an increasingly smaller part of the total traveled distance. Conversely, solving the Dubins TSP when the waypoints are close together compared to the turning radius leads to "cluttered" solutions because the aircraft will have to make wide turns to "get back to" the waypoints. Within certain limits, this can be countered by lowering the forward speed of the aircraft to reduce the turning radius. We will therefore be using Dubins curves with varying radii in this work. At the same time we will also consider the heading associated with each waypoint. Thus, we present a Dubins curve based trajectory generation and waypoint planning with variable radii, which is based on a variable forward speed. The planning is based on a genetic algorithm that modifies the both the continuous heading variable as in [10] and also the target speed in each waypoint.

2 Methods

The varying radii in Dubins curves and the application of a genetic algorithm to search for the optimal path with time as the performance measure are two distinct methods that we will address in this section. First, we present the generation of point-to-point Dubins curves with variable radii. This is followed by the formulation of a genetic algorithm that optimizes over the combinatoric sequence (solving the TSP) along with optimizing the continuous heading and target speeds of the waypoints.

2.1 Variable Speed Dubins Vehicle

The Dubins Curves that we will present here is an extension of the constant speed, bounded angular speed vehicle to a bounded speed, bounded angular speed vehicle. The forward speed v will be bounded

$$v_{\min} \leq v \leq v_{\max} \ , \tag{F.1}$$

and is subject to an acceleration that may be either $\pm a_{\max}$ or 0. It is assumed that the vehicle is able to turn with an angular velocity ω, which may take on only maximal values $\pm\omega_{\max}$ or 0. The forward speed and the angular speed defines the turning radius

$$r = \frac{v}{\omega} \ . \tag{F.2}$$

Further, it is assumed that the vehicle is only able to accelerate on straight stretches where the angular velocity is 0.

2.2 Point-to-Point Variable Radii Dubins Curve

We want to determine the length of a Dubins curve between two waypoints given the heading and speed in both waypoints. There are six possible combinations of segments forming a Dubins curve

$$\begin{gathered}\{\text{Right-Straight-Right}\}\ \{\text{LSL}\}\ \{\text{RSL}\} \\ \{\text{LSR}\}\ \{\text{RLR}\}\ \{\text{LRL}\}\ .\end{gathered}$$

The length of each of the three segments are denoted t, p, and q for the first, middle, and last segments, respectively. The heading is any real number between 0 and 2π, and the speed is any positive, real number, and defines the radius of the circle segment associated with the waypoint.

In [11], Shkel and Lumelsky derive formulas for computing the lengths of the different Dubins curve types along with a classification method to select the right type before the actual computation, this is opposed to the 'traditional' compute-and-compare approach. The approach in [11] is to translate, rotate, and scale the problem into canonical form

before computation. The canonical form places the initial waypoint in the origin, is scaled with the reciprocal of the turning radius, and rotated to place the final waypoint on the x axis.

The same idea of transforming the original problem into a canonical form will be used here. However, since the radii of are not equal in our case, a slightly different transformation is appropriate. In stead of transforming the problem to put the waypoints on the x axis, it is transformed to put the two centers of rotation on this axis. Note that this means that different transformations must be applied for the different path types.

We want to travel from one waypoint at $\mathbf{w}_1 = (x_1, y_1)$ with heading ϕ_1 and radius r_1 to another waypoint at $\mathbf{w}_2 = (x_2, y_2)$ with heading ϕ_2 and radius r_2. The centers of rotation then become

$$\mathbf{c}_{\text{right},i} = \mathbf{w}_i + r_i \begin{pmatrix} \sin(\phi_i) \\ -\cos(\phi_i) \end{pmatrix} , \tag{F.3}$$

$$\mathbf{c}_{\text{left},i} = \mathbf{w}_i - r_i \begin{pmatrix} \sin(\phi_i) \\ -\cos(\phi_i) \end{pmatrix} . \tag{F.4}$$

These parameters are all illustrated on Fig. F.1 for the case where the Dubins curve is made as Right-Straight-Right. The task now is to determine the lengths t, p, and q of the three segments, and for this we will use the transformation described above. The rotation angle is given by the angle $-\Theta$, the scaling by $1/r_1$, and the translation by $-\mathbf{c}_1$. This will transform the original configuration to the one seen in Fig. F.2. Now define the scaled radius of the second circle $\rho = r_2/r_1$ and the scaled distance between circle centers $d = \|\mathbf{c}_1 - \mathbf{c}_2\|/r_1$. The length of the short leg of the right triangle (gray in the figure) is $1 - \rho$ and the length of the hypotenuse is d. Thus, the angle at which the straight segment intersects the two circle is given as $\phi = \arccos\big((1-\rho)/d\big)$ and the length of this same segment is $p' = \sqrt{d^2 - (1-\rho)^2}$. Now, finally, let α_i be the clockwise angle between the x axis and $\mathbf{w}_i - \mathbf{c}_i$, which is given as $\alpha_i = \phi_i - \Theta + \pi/2$. Then $t' = \alpha_1 - \phi$ and $q' = \rho(\phi - \alpha_2)$ (constrained to the interval 0 to 2π). For this to make sense we need only require that $d \geq 1 - \rho$, meaning that none of the circles are properly inscribed in the other (in which case no tangent exists between the two).

Extending this result to include also RSL, LSL, and LSR we get

$$p' = d\sqrt{1 - \lambda^2} \tag{F.5}$$

$$t' = \begin{cases} (\alpha_1 - \phi) \mod 2\pi & \text{RSR, RSL} \\ (\phi - \alpha_1) \mod 2\pi & \text{LSL, LSR} \end{cases} \tag{F.6}$$

$$q' = \begin{cases} \rho((\phi - \alpha_2) \mod 2\pi) & \text{RSR} \\ \rho((\alpha_2 - \phi) \mod 2\pi) & \text{LSL} \\ \rho((\phi - \alpha_2 - \pi) \mod 2\pi) & \text{LSR} \\ \rho((\alpha_2 - \phi - \pi) \mod 2\pi) & \text{RSL} \end{cases} \tag{F.7}$$

where

$$\alpha_1 = \begin{cases} \phi_1 - \Theta + \frac{\pi}{2} & \text{RSR, RSL} \\ \phi_1 - \Theta - \frac{\pi}{2} & \text{LSL, LSR} \end{cases}$$

$$\alpha_2 = \begin{cases} \phi_2 - \Theta + \frac{\pi}{2} & \text{RSR, LSR} \\ \phi_2 - \Theta - \frac{\pi}{2} & \text{LSL, RSL} \end{cases}$$

$$\phi = \begin{cases} -\arccos\lambda & \text{LSL, LSR} \\ \arccos\lambda & \text{RSR, RSL} \end{cases}$$

$$\lambda = \begin{cases} (1-\rho)/d & \text{RSR, LSL} \\ (1+\rho)/d & \text{LSR, RSL} \end{cases}$$

$$d = \frac{\|\mathbf{c}_1 - \mathbf{c}_2\|}{r_1} \qquad \rho = \frac{r_2}{r_1}$$

$$d \geq \begin{cases} 1-\rho & \text{RSR, LSL} \\ 1+\rho & \text{RSL, LSR} \end{cases}$$

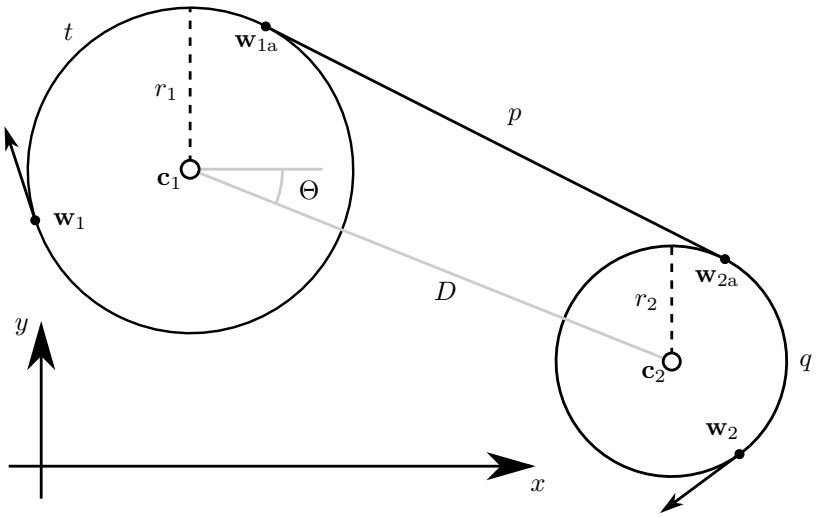


Fig. F.1: Dubins curve in original form.

As the lengths are both rotation and translation invariant, the original lengths $\{t, p, q\}$ are achieved by scaling $\{t', p', q'\}$ by r_1.

Since we do not allow acceleration of the vehicle on the curving part of the path the CCC cases are of limited used, and therefore these cases are not included in this work.

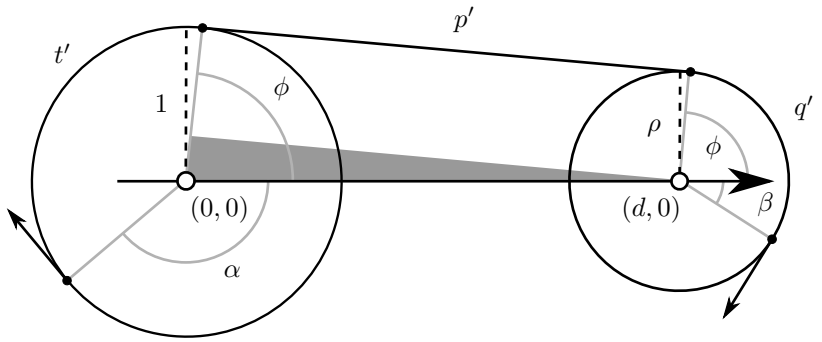


Fig. F.2: Canonical representation of the path seen in Fig. F.1.

2.3 Interpolation functions for Dubins Curves

Given the lengths of the non-canonical segments, three motion operators will translate a point along the path. Let $C : \mathbf{R}^5 \mapsto \mathbf{R}^3$ be a mapping along a circle. The input is $[x\ y\ \phi\ v\ r]^\top$ where x, y is the initial position in $\mathbf{R}^2$, ϕ is the heading angle, v is the arc distance and r is circle radius (which should be positive to make left turns and negative to make right turns). Then C is given by

$$C([x\ y\ \phi\ v\ r]^\top) = \begin{bmatrix} x - r\left(\sin(\phi) - \sin\left(\phi + \frac{v}{r}\right)\right) \\ y + r\left(\cos(\phi) - \cos\left(\phi + \frac{v}{r}\right)\right) \\ \phi + \frac{v}{r} \end{bmatrix} .$$

A translation along the straight segment is more simple, and is given by $S : \mathbf{R}^4 \mapsto \mathbf{R}^3$ as

$$S([x\ y\ \phi\ v]^\top) = \begin{bmatrix} x + v\cos(\phi) \\ y - v\sin(\phi) \\ \phi \end{bmatrix} .$$

Note that ϕ in S is the heading of the straight segment. If the vehicle starts at waypoint $\mathbf{w}_1$ with heading ϕ_1 and is following a left turn circle of radius r_1 then a) the position of the vehicle when it leaves the circle to traverse the straight segment, and b) the position when it leaves the straight segment to enter the second (right turn) circle with radius

r_2, and c) the position when the entire maneuver is completed, are given by

$$\text{a)} \qquad \mathbf{w}_{1\text{a}} = C\left(\begin{bmatrix} x_1 \\ y_1 \\ \phi_1 \\ t \\ r_1 \end{bmatrix}\right)$$

$$\text{b)} \qquad \mathbf{w}_{2\text{a}} = S\left(\begin{bmatrix} \mathbf{w}_{1\text{a}} \\ p \end{bmatrix}\right)$$

$$\text{c)} \qquad \mathbf{w}_2 = C\left(\begin{bmatrix} \mathbf{w}_{2\text{a}} \\ q \\ -r_2 \end{bmatrix}\right)$$

These functions may be useful when visualizing the curves of a computed path.

2.4 Velocities for Time-Optimal Traversal of the Curves

The straight segments are the only places where the vehicle is allowed to accelerate to match the initial velocity v_1 in (x_1, y_1, ϕ_1) with the final velocity v_2 in (x_2, y_2, ϕ_2). The time-optimal traversal of the segment is achieved by moving as fast as possible. This means that the vehicle must accelerate from v_1 to maximum velocity and later reduce it to v_2.

Assuming first that there is no upper limit on the velocity, then on the segment p the maximum velocity that can be achieved v_m, subject to a maximum allowed acceleration of a, must satisfy

$$p = \big(v_1 + \frac{v_\text{m} - v_1}{2}\big)\tau_1 + \big(v_2 + \frac{v_\text{m} - v_2}{2}\big)\tau_2$$

where $\tau_k = (v_\text{m} - v_k)/a$ are the times it takes to reach v_m from v_1 $(k = 1)$ and the time it takes to reach v_2 from v_m $(k = 2)$, respectively. Solving this gives

$$v_\text{m} = \sqrt{\frac{2pa + v_1^2 + v_2^2}{2}}\,. \tag{F.8}$$

Now, if this velocity is higher than the maximum allowable velocity v_max then we simply impose this limit and the total travel time on segment p becomes

$$\begin{aligned} \tau_p &= \tau_1 + \tau_2 + [\text{delay due to speed limit}] \\ &= \frac{v_\text{m} - v_1}{a} + \frac{v_\text{m} - v_2}{a} + \frac{(v_\text{m} - v_\text{max})^2}{a v_\text{max}}\,. \end{aligned}$$

In case the velocity limit is not reached the third term is (defined to be) zero. The time for traversal of the entire path is then

$$T = \tau_t + \tau_p + \tau_q = \frac{t}{v_1} + \tau_p + \frac{q}{v_2}\,. \tag{F.9}$$

Because of the bounded acceleration the path may be infeasible if the difference between the initial and final speeds is too large to realize over the p-segment. When this is the case, $v_{\mathrm{m}} < \max\{v_1, v_2\}$.

2.5 Genetic Algorithm

To find the fastest path through the waypoints we employ a genetic algorithm. In this case there are three parameters that determine the fitness of the solution when varied. The parameters are the sequence of the waypoints, the heading in each waypoint, and the velocity in each waypoint. In the traditional case where the Euclidean distance between waypoints is used the choice of connecting edges in the TSP graph are decoupled resulting in a entirely combinatorial problem. However, in the Dubins TSP case, the headings of the waypoints couples the segments so that each Dubins path cannot be optimized separately from the order of the waypoints. The same applies to the velocity in this variable-speed Dubins TSP.

Much work has been done in the genetic algorithms for Euclidean TSP, for a review see [12]. Yu and Hung [10] extends some of the traditional methods for use in Dubins TSP. Much of their representation is used here to adapt the genetic algorithm to the specific problem at hand.

Encoding

The path representation, described in [12], encodes the problem in an ordered list of indices to the waypoints. In the jargon of genetics, the encoded solution to the problem is called the genome and the entries of the genome is called the genes. This way, the ordered list of indices is the genome and each index is a gene. Yu and Hung [10] extends the path representation to include the heading in each gene. The same approach is taken here; extending the representation with the velocity, so that a genome is a sequence of genes of the form (i, ϕ, v). The genome example

$$\{(1, 1.4, 2.3), (3, 3.6, 2.7), (2, 4.1, 1.6), (4, 1.1, 1.2)\}$$

starts in the waypoint indexed 1 with the heading 1.4 radians and the velocity 2.3, and move through waypoints 3, 2, and 4 with their associated headings and velocities.

Fitness

The fitness function evaluates how fit each individual in the population is. As the objective is to minimize the time spent, individuals with a lower time cost should have a higher fitness than slower individuals. This is achieved here by letting the fitness equal to the time of the slowest genome of the population minus the time of the evaluated genome.

```
Require: N_p, P_c, P_inv, P_ex, P_dis, P_φ, P_v
  procedure GENETIC ALGORITHM
      Initialize population ← N_p random genomes
      doTerminate ← False
      while not doTerminate do
          newPopulation ← empty population
          Make ROULETTEWHEEL selector from population
          for i ← 1, N_p do
              repeat
                  if RAND(0, 1) ≤ P_c then
                      parent_1 ← ROULETTEWHEEL
                      parent_2 ← ROULETTEWHEEL
                      child ← CROSSOVER(parent_1, parent_2)
                  else
                      child ← ROULETTEWHEEL
                  end if
                  if RAND(0, 1) ≤ P_inv then
                      child ← INVERSE(child)
                  end if
                  if RAND(0, 1) ≤ P_ex then
                      child ← EXCHANGE(child)
                  end if
                  if RAND(0, 1) ≤ P_dis then
                      child ← DISPLACE(child)
                  end if
                  if RAND(0, 1) ≤ P_φ then
                      child ← φ-RANDOM(child)
                  end if
                  if RAND(0, 1) ≤ P_v then
                      child ← v-RANDOM(child)
                  end if
                  Add child to newPopulation
              until VALID(child)
          end for
          Evaluate newPopulation
          if bestNewGenome > allTimeBest then
              allTimeBest ← bestNewGenome
          else
              worstNewGenome ← allTimeBest
          end if
          population ← newPopulation
          doTerminate ← evaluate termination criterion
      end while
  end procedure
```

Fig. F.3: The algorithm for the genetic algorithm.

Selection

A roulette wheel selection is used. Here each bin of the roulette wheel has a width equal to the fitness of the corresponding individual. This way the roulette ball selects more fit individuals more often.

Crossover

The crossover operator combines parts from two parent genomes to produce an offspring. In this work, the order crossover [13] is used. This crossover tries to combine segments of the parents while conserving their order.

The procedure is to choose a subsection from one parent, which is copied to the offspring, and then fill in the blanks with the sequence from the other parent less the already used indices. For example, if the genomes

$$\{1, 2, 3, 4, 5, 6, 7\} \qquad \text{and} \qquad \{4, 6, 2, 7, 1, 3, 5\}$$

are combined, choosing the subsection $\{3, 4, 5\}$ from the first genome, they will produce the offspring

$$\{6, 2, 3, 4, 5, 7, 1\} \, .$$

Mutation operators

Mutation operators operates on a single genome, modifying the representation in the hope that this may produce a more fit individual. Yu and Hung [10] adopt and extend the inversion and exchange mutations and introduce the shift mutation, which modifies the continuous heading value. The method of their shift mutation is used here to randomly choose headings and velocities. Here, the mutations are called ϕ-random and v-random. The extended inversion mutation of Yu and Hung is adopted here along with their shift mutation, whereas the traditional exchange is used instead of their extension. Furthermore the traditional displace mutation is also used.

The individual mutations are shortly described here. For a more in-depth review of mutation- and crossover-operators, see [12].

The inversion mutation chooses a subsection of the mutatee and replaces it with itself reversed, the extended version used here also shifts the heading of each of the affected indices by π, reversing the direction of travel.

The exchange mutation simply chooses two genes and exchanges them. Here, the heading is not shifted as proposed in [10].

The displace mutation chooses a subsection of the genome and moves it forwards or backwards in the sequence, also here, the heading is untouched.

The random mutations chooses a gene and sets the continuous values to a random value. The ϕ-random sets the heading to a value in $[0, 2\pi[$, and the v-random set the velocity to a value in $[v_{\min}, v_{\max}]$.

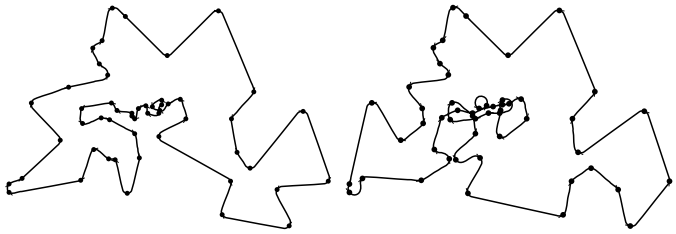


Fig. F.4: Two instances of Dubins genetic algorithm solutions to the Berlin52 problem corresponding to the results listed in Table F.1. Left: Minimum initialized. Right: Maximum initialized.

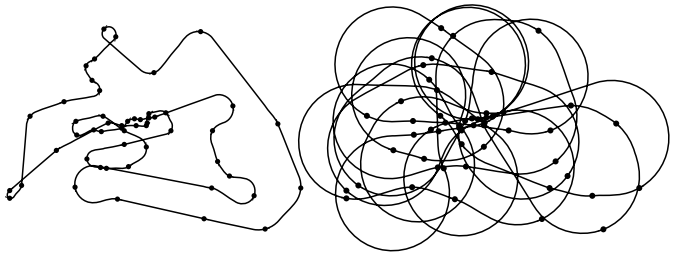


Fig. F.5: Two instances of Dubins genetic algorithm solutions to the Berlin52 problem corresponding to the results listed in Table F.2. Left: Minimum initialized. Right: Maximum initialized.

3 Results

The Berlin52 problem from the TSPLIB package [14] serves here as demonstration. It contains 52 locations of interest in Berlin, Germany, spread out in the interval $[(0,0),(1740,1175)]$ with about 20 of the points located relatively close around the center and the rest in the periphery. Our proposed method is well suited to this problem because it has both widely spaced and closely spaced waypoints. The traditional Dubins methods with fixed turning radius cannot adapt to the need for both slow and fast turns to accommodate the close and widely spaced waypoints.

3.1 Simulation results

We have examined two instances of a genetic algorithm optimization. One where the maximum speed is low and one where it is high. The results presented in Table F.1 are for an instance where the possible speeds are in the interval $[0,100]$ units per second with an acceleration of 10 units per second squared and rotational speed of 3 radians per second. In Table F.2, the results are given for a setup with the maximum speed and acceleration increased by a factor 10.

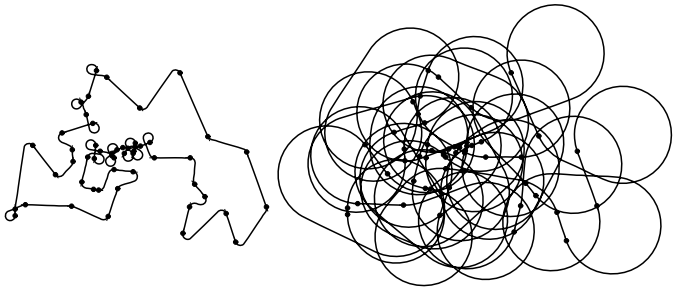


Fig. F.6: Two examples of Alternating Algorithm–Euclidean genetic algorithm solutions to the Berlin52 problem, left and right corresponding to the instances in Table F.1 and Table F.2 respectively.

This result is compared to solutions obtained by the Alternating Algorithm (AA) of Savla et al. [2]. Such a solution consists of two steps; 1) find the desired order of the waypoints (we use both a Nearest Neighbor and a Euclidean GA), and 2) set the headings of the waypoints in pairs so that the segments connecting the waypoints alternates between a line segment and a Dubins path. The fitness is measured in the same way as for the Dubins GA, that is, acceleration to maximum allowable speed is used on any straight segment, while turns have a constant velocity linearly proportional to the turning radius. Two examples of AA solutions are shown in Fig. F.6. Here the sequence of waypoints are found using the genetic algorithm with a Euclidean distance cost function. To allow a fair comparison, several instances of the AA solution is evaluated, varying the (fixed) forward speed in the turns from the minimum to the maximum speed, so that the turning radii varies from least to largest. In both examples the maximum speed yielded the fastest solution. Note that setting the forward speed in the turns to zero reduces the turning radii to nothing and equates to a Euclidean solution where the vehicle slows down to a stop, rotates on the spot (i.e. center turns like a tank or hovering helicopter), and speeds up along the next line section towards the next point.

While the first instance with low maximum speed and acceleration gives reasonable results for the Euclidean GA with AA, it has a bit of trouble in the crowded center. The second instance has a turning radius of 333.3 units at the speed of 1000 units per second, which is a lot compared to the individual distances between the points. This large turning radius proves difficult and gives considerable trouble in the entire region. The solutions can be seen in Fig. F.6.

Six methods are listed in table F.1. First the nearest neighbor heuristic solution with an AA smoothing with a forward speed of 0 (center turns) and next with a forward speed of 100, which translates to turns with radius 33.3. The next two rows are also smoothed with the AA with forward speeds of 0 and 100 respectively, but here the underlying

Table F.1: Costs for solutions to the Berlin52 problem.

Method	Length	Time	
Nearest Neighbor + 0 AA	8,980.9	541.7	
Nearest Neighbor + 100 AA	11,796.3	118.0	
Euclidean GA + 0 AA	7,835.7	501.8	
Euclidean GA + 100 AA	10,607.0	106.1	*Fig. F.6*
Dubins GA (min. init.)	8,476.1	102.2	*Fig. F.4*
Dubins GA (max. init.)	9,254.3	92.6	*Fig. F.4*

Speed: [0, 100], Acceleration: 10, Rotational Speed: 3

solution is obtained with a genetic algorithm (GA) based on a Euclidean cost function. Lastly, two instances of the Dubins GA are listed. The difference between the two is the way of initialization. The first (minimum initialization) is initialized with a population of identical solutions obtained with the nearest neighbor heuristic, setting the speed and headings of the waypoints to 0. The next entry is initialized just like the other, but with the speeds set to maximum.

All the solutions are evaluated according to two metrics. The first is the traveled distance, i.e. length of the solution, the next is the time it takes to traverse it. Note that the Euclidean GA uses the Euclidean distance as the cost function for optimization, but the distance listed in the table is the distance after AA smoothing. In table F.2 the results of the same instances, but with a higher maximum speed and acceleration are listed.

Table F.2: Costs for solutions to the Berlin52 problem.

Method	Length	Time	
Euclidean GA + 0 AA	7,835.7	246.9	
Euclidean GA + 1,000 AA	73,125.8	73.1	*Fig. F.6*
Dubins GA (min. init.)	10,732.7	50.9	*Fig. F.5*
Dubins GA (max. init.)	39,091.6	39.1	*Fig. F.5*

Speed: [0, 1,000], Acceleration: 100, Rotational Speed: 3

The computational effort of the Dubins GA is considerable compared to the Euclidean GA. The instances of the Euclidean GA were solved in 10,000 generations, where the Dubins GA required 15,000 generations, but because of the more computationally intensive cost function of the Dubins curves, the run time of the Dubins GA

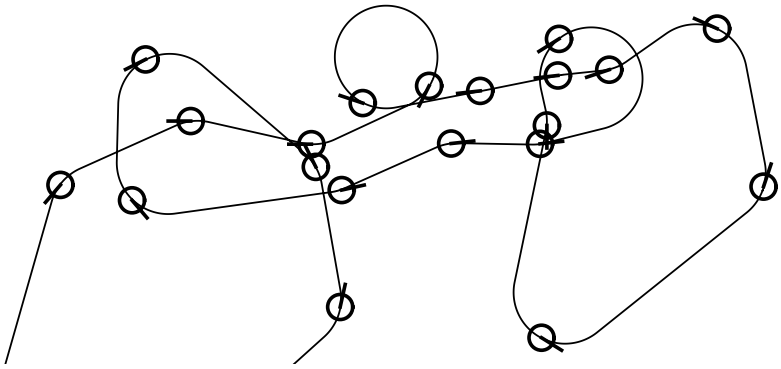


Fig. F.7: Close-up of the center section of the maximum initialization instance in Fig. F.4.

was roughly six times as long. Specifically 115 seconds for the Dubins GA compared to 20 seconds for the Euclidean GA on a Intel Core i7 2.80 GHz core.

3.2 Observations

When examining the results, note that the heading of the waypoints tend to a value, so that the circle segments on either side are of equal length, and further that no waypoint has a heading that results in the path having a left-turn when entering the waypoint and a right-turn exiting (or vice versa). See a close-up of the center section of Fig. F.4 in Fig. F.7 with the headings of the waypoints illustrated.

4 Discussion

In all instances, the Dubins genetic algorithm outperforms the other solutions by matter of time. Interestingly, the method of initializing the algorithm has a strong influence on the obtained solution. The nearest neighbor solutions in Table F.1, gives some insight to this behavior. The solution initialized with minimum speed produces a short, but slow execution where the maximum speed initialization produces a longer, but faster execution. As the algorithm incorporates a validity check, all offspring with too large difference between the initial and terminal speeds to be realized are discarded. Thus the algorithm may obtain a solution that is trapped in a local minimum, which the mutations are not “strong” enough to break out of.

A possible way to visualize this problem is by thinking of the solution space as a hill with the best solution sitting at the top of the hill. As the solutions are mutated, they may move a step up or down the hill. The selective pressure of the genetic algorithm favors the solutions that are stepping up the hill. The validity check creates boundaries

on the hill that the solutions cannot step over, this could be thought of a a form of ravine dividing the faces of the hill. The solution may get stuck on one side of the ravine in a local maximum and have to cross the ravine to get to the global maximum on the other side. If the mutation operators are not "strong" enough to step across the ravine in one step, the solution have to backtrack down the hill to a point where the ravine becomes so narrow that it can be negotiated. The genetic algorithm allows a portion of backtracking, but the probability dwindles as more steps are needed.

This is what seems to happen in the case of maximum- and minimum-initialized initialization. The ordering of the waypoints and their headings are fairly quick at converging to a nice solution, where the speed adaption process are more slow. The ravine is thus created as new changes in heading or ordering may easily become infeasible because of a too large speed. Thus, the solution have to backtrack by first reducing the speed, then changing the ordering, and finally the heading. The probability of this sequence of mutations happening is quite low.

There are different ways to overcome the problem of getting caught in local minima. Here, one of the problems is that the algorithm cannot search from one feasible domain through an infeasible domain to another feasible domain. Such problems are addressed by Ray et al. in [15], where they allow a portion of infeasible solutions in the population. They also note that optimal solutions often lie on the constraint boundaries, an observation that may be seen in this case as well: The maximum-initialized version maintains the maximum speed throughout the entire path, thus lying on the speed constraint boundary. Conversely, the minimum-initialized version moves on the acceleration constraint boundary; the reason why the speed does not reach the maximum. The approach by Ray et al. bridges or narrows the ravines, easing the transition from one feasible domain to another.

Interestingly, the sub-optimal minimum-initialized algorithm generally produces "pleasingly" looking paths, whereas the maximum-initialized paths tend to look more complex (although faster). Rather than dismissing the minimum-initialized approach we will regard it as a conservative version of the algorithm, where the maximum-initialized can be regarded as the aggressive version. The difference between the two is very apparent in Fig. F.5 corresponding to the solutions in Table F.2.

5 Further Work

The observation that the circle segments seem to be divided on the middle seems intuitive as this results in the shortest straight segments, which definitely minimizes the length. But with the acceleration and speed constraints, the same conclusion is not so obvious. If a closed form solution to this is constructed, there would be no reason for the genetic algorithm to individually optimize the headings as they would be an implicit consequence of the ordering and speed.

Even if no closed form is found, the observation may be used as a heuristic when constructing initial solutions for the genetic algorithm.

In this case, the vehicle is constrained to only accelerating on the straight segments. For a real aircraft, however, it is possible to accelerate during a turn. As described in (F.2) the radius is defined by the forward and angular speed of the aircraft. If the aircraft is to accelerate in the turns, the radius will become a function of the speed, and the path will become a spiral segment. Planning with such spirals in stead of circle segments would certainly bring the model closer to the real system, and construct even better trajectories. Spirals have been used in planning for aircraft. One often used is the clothoid, which has a curvature that is linear with arc length [16]. While the clothoid is an approximation of the spiral alluded to here, it models a constant forward speed and a bounded angular acceleration.

The differences between the two solutions from the maximum- and minimum-initialized algorithms show that there may be other factors in the optimization of this problem than simply the time consumption. The minimal-initialized solution definitely looks more pleasing to the human eye, even though it is slower. Such a factor may very well be important when motion planning for robots in human environments. This may lead to the use of a multi-objective optimization instead; with both the time and distance as optimization factors. In [17], a genetic algorithm have been developed to handle such problems.

Another approach to this multi-optimization problem is to consider the fuel usage of the trajectory. Such a parameter will combine the two other parameters, as it will strike a balance between the fastest solution and the slowest solution, choosing a speed that is probably closer to the cruising speed of the aircraft, where it has the best mileage.

Acknowledgment

This work is supported by the Danish Council for Strategic Research under grant no. 09-067027 (ASETA). See `www.aseta.dk` for more details.

References

[1] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, no. 3, p. 497, Jul. 1957.

[2] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins' vehicle," in *Proceedings of the 2005, American Control Conference, 2005.* Portland, OR, USA: IEEE, 2005, pp. 786–791.

[3] J. L. Ny, E. Feron, and E. Frazzoli, "On the Dubins Traveling Salesman Problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2012.

[4] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2379–2384.

[5] S. Hota and D. Ghose, "Optimal geometrical path in 3D with curvature constraint," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2010, pp. 113–118.

[6] P. Sujit, S. Saripalli, and J. Sousa, "Unmanned Aerial Vehicle Path Following: A Survey and Analysis of Algorithms for Fixed-Wing Unmanned Aerial Vehicles," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, 2014.

[7] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum Press, 1972, pp. 85–103.

[8] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[9] K. Helsgaun, "General k-opt submoves for the Lin–Kernighan TSP heuristic," *Mathematical Programming Computation*, vol. 1, no. 2-3, pp. 119–163, Jul. 2009.

[10] X. Yu and J. Y. Hung, "A genetic algorithm for the Dubins Traveling Salesman Problem," in *2012 IEEE International Symposium on Industrial Electronics*. IEEE, May 2012, pp. 1256–1261.

[11] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, Mar. 2001.

[12] P. Larrañaga, C. M. Kuijpers, R. H. Murga, I. n. Inza, and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[13] L. Davis, "Applying adaptive algorithms to epistatic domains," in *IJCAI'85 Proceedings of the 9th international joint conference on Artificial intelligence*, vol. 1. Morgan Kaufmann Publishers Inc., Aug. 1985, pp. 162–164.

[14] G. Reinelt, "TSPLIB–A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, Nov. 1991.

[15] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, "Infeasibility Driven Evolutionary Algorithm for Constrained Optimization," in *Constraint-Handling in Evolutionary Optimization*, E. Mezura-Montes, Ed. Springer Berlin Heidelberg, 2009, pp. 145–165.

[16] A. Tsourdos, B. White, and M. Shanmugavel, *Wiley: Cooperative Path Planning of Unmanned Aerial Vehicles*. John Wiley & Sons, Ltd, 2011. [Online]. Available: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470741295.html

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.