Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Adatgyűjtő multiágensű rendszer pályatervezése érzékelő csomópontok között

TDK dolgozat

Készítette:

Szénási Sára

Konzulens:

Dr. Harmati István

2022

Tartalomjegyzék

Kivonat

Manapság egyre gyakoribb, hogy egy adott területről folyamatos vagy egyszeri adatgyűjtésre van szükség. Periodikus adatgyűjtésre például egy őrzött területen behatoló mozgásának feltérképezése, egy erdőben a vadak követése, vagy egy mezőgazdasági területen az időjárási elemek megfigyelése. Egyszeri adatgyűjtésre van szükség például egy katasztrófa sújtott, vagy háborús területen a túlélők felkutatása vagy a bajba jutott síelők megmentése során. Az adatgyűjtés vezeték nélküli szenzoros hálózatokkal (WSN) tehető meg a legegyszerűbben. Azonban a hálózaton belüli adatküldés nagy energia felhasználással jár, ami az akkumulátorról működő szenzorok élettartalmát, működési idejét jelentősen csökkentheti. Ezek cseréje gyakran problémás vagy nagy befektetést igényel. Ezért az adott érzékelő csomópontok közt az adatátvitelt robotokkal valósítom meg, ilyen módon nem kell távolra adatot küldeni, ami számottevően csökkenti az energiafelhasználást, és ennek következtében nagymértékben növeli a hálózat élettartalmát.

Az adatgyűjtés késleltetési idejének minimalizálása érdekében több együttműködő robot bevethető. Gyakori, hogy az egyes csomópontok meglátogatása sürgetőbb a többinél, például a súlyosabb sérültek hamarabb ellátásra szorulnak. Ebből adódóan prioritásos csomópontokról való adatgyűjtéshez is tervezek útvonalat.

A szenzoros mezőn a bázis és az érzékelő csomópontok helyzetét ismertnek tételezem fel. A periodikus útvonal tervezés megvalósításához olyan algoritmust fejlesztettem, amelynek során a robotok a bázis-csomópontból kiindulva együttesen az összes érzékelő-csomópontot bejárják, azokról az adatokat letöltik, majd a bázishoz visszaérve azokat feltöltik további feldolgozásra. Tehát a megtervezett útvonalon a robotok ismételten végig haladnak a folyamatos adatgyűjtés érdekében. Ekkor a feladat visszavezethető utazó ügynök problémára, illetve több szereplős utazó ügynök problémára, amelyet heurisztikus módszerrel, hangya kolónia algoritmussal oldok meg.

Az egyszeri adatgyűjtéshez általam kifejlesztett algoritmus működése során az ágensek meghatározott vagy véletlenszerű pontból indulnak ki, meglátogatják az összes szenzoros csomópontot onnan letöltik az adatokat majd pedig egy kijelölt pontba érkeznek meg ahova feltöltik a begyűjtött információt. Ekkor a cél egy olyan költség minimalizálása, amely a bejárási időn túl magába foglalja a magasabb prioritású csomópontok késői meglátogatásának a következményét is. Az optimalizálást ebben az esetben is hangya kolónia algoritmussal oldom meg.

A fejlesztett algoritmusokat Matlab környezetben implementáltam. Szimulációkat végeztem több különböző érzékelő mezőn, különböző számú ágenseket tartalmazó multiágensű rendszerekkel. Az adatgyűjtés hatékonyságát több különböző mérőszámmal értékeltem.

Abstract

Nowadays there is a more and more common need for continuous or one-off data collection on a specified area. Continuous data collection can be for example the tracing of the movements of an intruder on a protected area, the following of games in a forest, or the observation of weather on an agricultural area. One-off data collection is necessary in case of searching for the survivals of disasters or wars, or, for saving of skiers get into trouble. The simplest way of such data collection is using wireless sensor network (WSN). However, data transmission within the network uses great amount of energy that may reduce significantly the lifetime and operation time of the battery operated sensors. The replacement of batteries is often problematic or requires large investments. For this reason, the data transfer between the given sensor nodes is carried out with robots. This way there is no need to send data to remote destinations so the amount of energy used decreases significantly which increases the lifetime of the network.

In order to minimize the latency of the data collection it is possible to use more cooperating robots. It frequently occurs that one of the nodes requires more urgent visit than the other ones, for example the serious injuries requires urgent medical attendance. To solve this, path planning for data collection from priority nodes are also presented here.

The location of the base node and the sensor nodes on the sensor field are assumed to be known preliminary. In order to achieve periodic path planning an algorithm is developed where the robots visit all nodes starting from the base node, download the data from the nodes and uploads the collected data for further data procession after returning to the base node. So the robots will circuit on the planned path repeatedly in order to collect data continuously. The problem can be originated in the travelling salesman problem with one or more agents that are solved here using the ant colony heuristic algorithm.

The algorithm developed here to perform the one-off data collection operates so that the agents starts from preliminary determined or random points and then visit all the sensor nodes, download the data form these nodes and after this arrive to a designated point where upload the collected information. The goal here is the minimization of such a cost that include not only the walk-through time but also the consequences of the late visit of the higher priority nodes. Again, the optimization is also solved here using the ant colony algorithm.

The algorithms were implemented in MATLAB environment. Simulations were carried out on multiple different sensor fields using different number of agents in agent systems. The efficiency of data collection were evaluated with more different measures.

1. fejezet

Bevezetés

Manapság gyakran előfordul, hogy egy adott területen folyamatos vagy egyszeri adatgyűjtésre van szükség. Folyamatos adatgyűjtés szükséges például egy mezőgazdasági területen az időjárási elemek megfigyelésére, mint például hőmérséklet, csapadék mennyiség, páratartalom, a növények ellátási igényeinek felmérésére, vagy egy katonai őrzött területen az esetleges behatolók mozgásának feltérképezésére. Egyszeri adatgyűjtés szükséges például egy katasztrófa sújtott területen a bajba jutottak feltérképezésére. Ekkor a mentő expedíció során figyelembe kell venni a sérülések súlyosságát és a közvetlen környezetükből eredő további veszélyforrást. Tehát ekkor prioritásos csomópontok közötti útvonaltervezés szükséges.

Az adatgyűjtés vezeték nélküli szenzoros hálózatok (Wireless Sensor Network–WSN) segítségével zajlik a legtöbb esetben [1] [2]. A legtöbb felhasználás során a szenzoros hálózat két részből áll: a bázisból, ahova az adatok feltöltésre kerülnek és sok szenzoros csomópontból, amelyek a környezetből való adatgyűjtésért felelősek [3]. Az érzékelő csomópontok tipikusan akkumlátorról működnek, amelyeket sok alkalmazásban nem lehet kicserélni, például egy háborús területen [4]. Ezért fontos a csomópontok energia felhasználásának minimalizálása. A távoli adatküldés nagy energia disszipációval jár, hiszen az adatvesztési ráta az adat küldési távolsággal együtt nő és az adatátviteli ráta az energia fogyasztással arányos. Mivel a csomópontok akkumlátorról működnek a távoli adatküldés jelentősen csökkenti a hálózat élettartalmát. Ezért az adatátvitelt adatgyűjtő robotok segítségével hajtjuk végre [5] [6]. Az adatgyűjtő robotok segítségével az adatküldés kis távolságokra is elérhetővé válik, amely több előnyt is magában rejt. Csökkenti az adatátvitel energia igényét és egyúttal az átvitel hibaszázalékát és növeli a hálózat megbízhatóságát, amely több alkalmazási területen is fontos szempont [4]. A [7] egy áttekintő publikáció a mobilis robotok számos felhasználásának a vezeték nélküli szenzoros hálózatokban. További alkalmazás például repülő mobilis robotok összehangolt adatgyűjtése [8], vagy tömegközlekedéssel való adatgyűjtésre [9]. A [10] irodalomban Dubins robotok használatával valósítanak meg periodikus adatgyűjtést.

Az adatgyűjtést omnidirekcionális robotokból álló multiágensű rendszerrel hajtom végre. Az omnidirekcionális robotoknak számos különböző kialakítása ismert, ahogy azt [11] irodalom összefoglalja. Fontos tulajdonságuk a holonomitás: bármilyen útvonalon végig tudnak haladni, egyszerre tudnak transzlációs és rotációs mozgást végezni. Ez jelentősen megkönnyíti az útvonaltervezést, és csökkenti a bejárási időt. A megoldás során feltételezzük, hogy az adatletöltés nem pillanatszerű, hanem az függ az adatmennyiségtől, az adatátviteli sebességtől és a csomóponttól való távolságtól. A robot csak akkor vesz fel adatot, ha közvetlenül a csomópont alatt helyezkedik el, és adatletöltés közben a pozícióját nem változtatja a megfelelő idejű és minőségű adatletöltés érdekében. A robot akkor is tudna adatot letölteni, ha a csomópont körüli bizonyos tartományon belül mozog, azonban

ekkor csökkenne az adatátvitel gyorsasága és minősége is. Ráadásul a közelebbi adatcsere kisebb energia felhasználással jár, és biztonságos adatátvitelt biztosít. A csomóponton felhalmozott adat mennyisége és az adatátviteli sebesség együttesen határozza meg a szükséges adatletöltési időt. Jelölje g_i az i csomóponton felhalmozott adatmennyiségét. Ekkor a szükséges kontakt idő δ_i a következő módon számítható:

$$\delta_i = \frac{g_i}{r(h_i)}, \tag{1.1}$$

ahol h_i jelöli a robot és a csomópont távolságát, abban az esetben, ha a robot közvetlenül a csomópont alatt helyezkedik el. Az adatátviteli ráta $r(d)$ függ a csomópontoktól való távolságtól, annak egy nemnövekvő függvénye a lépcsőház model [12] szerint: $r(d)$ (bit/s):

$$r(d) = \begin{cases} r_1,\ 0 < d \leqslant d_1 \\ r_2,\ d_1 < d \leqslant d_2 \\ \vdots \\ r_i,\ d_{i-1} < d \leqslant d_i \\ \vdots \\ 0,\ d > d_{max} \end{cases} \tag{1.2}$$

ahol d_{max} az a maximális távolság, ahonnan még a csomópontról adatot lehet letölteni.

Az adatgyűjtés során a robotoknak be kell járniuk az összes érzékelő csomópontot, le kell tölteniük az ott tárolt adatokat, majd a bázisállomásra érve fel kell tölteniük azokat. A robotok útvonalának a megtervezése egy fontos pont, hiszen az útvonal hossza közvetlenül befolyásolja az adatok késleltetési idejét.

A dolgozatomban először multiágensű robotrendszer számára tervezek útvonalat periodikus adatgyűjtés végrehajtásához. Ekkor az érzékelő mezőn elhelyezkedik egy bázis-csomópont ahonnan a robotok kiindulnak és ahova az adatok begyűjtése után vissza kell térniük és a begyűjtött adatokat fel kell tölteniük. Tehát ekkor az ágensek együttesen járják be az érzékelő mező csomópontjait folyamatosan. A cél a bejárási idő minimalizálása. Ez a probléma visszavezethető tehát a több ágenses utazó ügynök problémára (Multiple-Travelling Salesman Problem), amely NP nehéz probléma. Megoldására a heurisztikus hangya kolónia optimalizációt alkalmazom, amellyel számos irodalom foglalkozik [13] [14] [15][16][17][18].

Előfordulhat azonban, hogy az egyes csomópontokról való adatok begyűjtése nem egyforma prioritással rendelkezik, például egy katasztrófa súlytott területen a különböző sérülést szenvedők ellátása nem egyformán sürgős. Ekkor olyan algoritmusra van szükség amely egyensúlyt teremt a lefedett terület bejárási idejének minimalizálása és a kiemelt csomópontok minél előbbi meglátogatása között [19] [20] [21] [22]. A dolgozatomban megoldott feladat során a robotok a terep különböző pontjairól indulnak ki, meglátogatják együttesen az érzékelő csomópontok mindegyikét, majd pedig a bázispontba érve feltöltik a begyűjtött adatokat. Ez szintén egy NP nehéz probléma, amelyet ACO algoritmussal [23] [24] oldok meg.

A dolgozatom következő fejezetében bemutatom a periodikus útvonaltervezés lépéseit először egy robotra majd pedig multiágensű robotrendszer számára. A 3. fejezetben vázolom a prioritásos csomópontok közötti útvonaltervezést. A 4. fejezetben bemutatom a szimulációs eredményeket több különböző érzékelő mezőn, különböző számú robotból álló multiágensű rendszert alkalmazva. Az eredmények értékeléséhez mérőszámokat definiálok mind a periodikus adatgyűjtés [25], mind a prioritásos adatgyűjtés estében. Dolgozatomat az általam fejlesztett algoritmusok közvetlen felhasználásával zárom az 5. Kitekintés fejezetben, ahol az akadályokkal bővített érzékelési térben vázolom fel az útvonalterve-

zés lehetőségeit. A 7 fejezet tartalmazza a szimulációs eredményeket szemléltető videók elérhetőségét.

2. fejezet

Útvonal tervezés periodikus adatgyűjtéshez

A fejezetben először a periodikus adatgyűjtésről írok általánosságban, kifejtem ennek hasznosságát és a tervezés lehetőségeit. Majd bemutatom az általam készített algoritmust egy ágens útvonaltervezésére majd pedig általánosítom multiágensű rendszer számára.

A periodikus adatgyűjtés biztosítja a folyamatos megfigyelést egy adott területen. Ekkor a cél egy kijelölt báziscsomópontból kiindulva az összes érzékelő csomópont bejárása onnan az adatok letöltése majd a bázis csomópontba visszatérve a begyűjtött adatok feltöltése. Tehát ilyenkor egy zárt útvonal tervezése a cél. Ha a csomópontokat egy gráf csúcsainak tekintjük a köztük lévő bejárható útvonalat pedig a gráf éleinek, ezek hosszát pedig az élek súlyának, akkor a probléma vissza vezethető az utazó ügynök probléma megoldására. Az utazó ügynök probléma (Travelling Salesman Problem – TSP) egy eladó napi útvonalát tervezi meg úgy, hogy a házaló ügynök a saját otthonából indul el, meglátogatja az aznapra kiírt vevők mindegyikét majd pedig hazatér. Ennek során takarékossági megfontolásból minimalizálja az útvonalát. Formalizálva: egy gráfban a legrövidebb, minden csúcspontot tartalmazó kört, Hamilton kört keresi. Azonban a TSP egy NP nehéz probléma ami azt jelenti, hogy egzakt megoldása nem, vagy csak nagyon hosszú idő alatt lehetséges. Hiszen ha n jelöli a csúcspontok számát akkor az összes sorrend megvizsgálása n számú esetet jelent ami már $n = 12$ esetében is 479001600 számú esetet jelent. Ha egy megoldás kiszámításához $t = 0.001s$ szükséges, az összes megoldás kiszámítása öt és fél napig tart. A nagy számítási kapacitás és idő következtében érdemes más megoldási módokhoz például heurisztikus módszerekhez folyamodni. A dolgozatomban hangya kolónia algoritmust alkalmazok.

A hangyák viselkedésének útvonaltervezésre való hasznosságát először Marco Dorigo publikálta 1991-ben [23] majd 1997-ben egy TSP megoldására finomított rendszert hozott létre [24]. Azóta az algoritmusnak sok változata, finomítása keletkezett, amelyeket a [27] irodalom összegez. A következőkben röviden összefoglalom a hangya kolónia optimalizálás algoritmus elkészítésének néhány lehetőségét, majd pedig bemutatom a saját algoritmusomat az utazó ügynök probléma megoldására.

A hangya kolónia algoritmus, Ant Colony Optimalizaliton (ACO) azon alapul, hogy a hangyák képesek megtalálni a legrövidebb útvonalat a hangyaboly és az élelem között. A hangyák az általuk megtett úton feromont bocsájtanak ki, maguk után feromon ösvényt hagyva, amelynek erőssége az idő múlásával csökken. A hangyák a más hangyák által az útra helyezett feromont érzékelik és döntéseiket (követendő irányt) az alapján választják ki. Az erősebb feromon ösvényt nagyobb valószínűséggel követik. A rövidebb útvonalon erősebb lesz a feromon szint, mivel kevesebb idő alatt járják végig ezt a hangyák. Tehát a

többi hangya is ezen az útvonalon fog végighaladni. Így képesek megtalálni a legrövidebb utat két pont között.

Az algoritmus során mesterséges hangyák építik fel az útvonalat. A mesterséges hangyák számára a cél a legkisebb költségű útvonal megtalálása. A mesterséges hangyák egymás között indirekt módon, mesterséges feromon segítségével kommunikálnak, mesterséges feromon ösvényt hoznak létre. Az útvonalak elkészítése után, kiszámításra kerül minden útra a költség függvény, és a feromon értékek ennek függvényében kerülnek frissítésre. A mesterséges hangyák rendelkeznek memóriával, tehát egy gráf bejárása során el tudják tárolni például, hogy mely csúcspontokat látogatták meg eddig. Esetlegesen rendelkeznek információval egy-egy él hosszáról, tehát nem csak a feromon értékek alapján tájékozódnak. Illetve a mesterséges hangyák feromon frissítése gyakran az út készítése közben (lokális feromon frissítés), egy él bejárása után, és a teljes út elkészítése után (globális feromon frissítés) is megtörténik. A mesterséges hangyák ilyen módon diszkrét időben élnek.

A hangya kolónia heurisztika alapvetően három fő lépésből áll. Első lépésben megalkotásra kerülnek a hangyák útvonalai, a feromon szintek frissítésre kerülnek az útvonalak alapján. Második lépés a feromon szint csökkentéséért felelős, ami növeli a konvergenciát. Harmadik lépésben az olyan algoritmusok kerülnek végrehajtásra, amelyek a hangyák által egyénileg nem kivitelezhetőek, például az elitista feromon frissítés. Ekkor a "legjobb" hangyák útvonalai alapján (például legrövidebb bejárt út) kerül frissítésre a feromon.

A hangya kolónia algoritmus alapja, hogy a frissülő feromon értékek alapján létrehozott megoldás konvergál az optimumhoz. Azonban, ha csak egy hangyának konstruálunk útvonalat a feromon mátrix frissítés után, a megoldás könnyen csak egy lokális minimum felé konvergál. Ezért érdemes több, egymással párhuzamosan működő hangyát használni, amelyek összességében jól fogják reprezentálni a valószínűségi eloszlást, így a globális optimum felé tud konvergálni az algoritmus.

A feromon frissítésének több módja ismert. Az egyik legkézenfekvőbb megoldásnak tűnik, hogy a feromon frissítést minden egyes él választása után hajtjuk végre, mint ahogy az a természetben is előfordul. Ilyenkor azonban minden hangya választása után frissülnek a feromon értékek, ami a valószínűségek folyamatos változása miatt ronthatja a konvergenciát. Lényegében ugyanaz a hatás, mintha csak egy hangya készítene útvonalat. Ennek két alapvető típusa ismert: Ant-density és Ant-quantity [23]. Az Ant-density minden hangya után amely választotta az adott élt egy Q mennyiségű feromont ad hozzá. Az Ant-quantity a Q mennyiséget elosztja az él d_{ij} hosszával, így a rövidebb élekre több feromon kerül. Annak érdekében, hogy elkerüljük, hogy a legerősebb, legrövidebb, élt az összes hangya válassza, ezen élek választása után közvetlenül végrehajthatunk egy büntető lokális feromon frissítést, amely csökkenti ezek választási lehetőségét [27]. Az egyik leggyakrabban használt megoldás, hogy az egyes hangyák útvonalának létrehozása után hajtunk végre feromon frissítést: Ant-cycle. Ekkor a feromon a teljes bejárt út után kerül frissítésre. Ilyenkor nemcsak a teljes bejárt út hosszát, hanem esetleg a teljes költségét is lehet használni alkalmazástól függően. Tehát ekkor esetleges más erőforrások költségei is beszámíthatók, például irányváltoztatáshoz szükséges út vagy idő. A globális feromon frissítés is népszerű, amikor ugyanis csak a legjobb hangyák útja után kerül frissítésre a feromon, elitista stratégia. Ezenkívül még számos feromon frissítés módszer ismert [27] ezek közül többet is tartalmaz. A különböző módszerek természetesen együttesen is használhatók a lehető legjobb megoldás elérése érdekében.

2.1. Az utazó ügynök probléma megoldása hangya kolónia algoritmus alkalmazásával

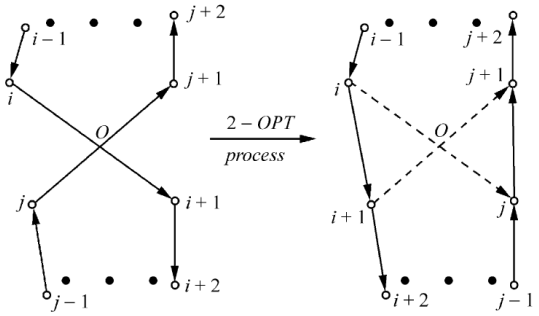
Az utazó ügynök probléma megoldása során a hangyák kezdetben egy véletlenszerűen választott csomópontból indulnak. Azonban a TSP azon kitétele miatt, hogy minden várost pontosan egyszer kell meglátogatni, minden hangyára elraktározandó, hogy mely csúcspontoknál járt már, ez az úgynevezett tabu list. Illetve csak akkor fejezhető be a készített út, ha már minden csúcspont bekerült a tabu list-be.

A csúcspontok kiválasztásának valószínűsége az i-edik csúcsból a j-edik csúcsba a k-adik hangya számára a feromon szintek és a távolság értékek segítségével határozható meg a következőképpen:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & \text{ha } j \in \mathcal{N}_i^k \\ p_{ij}^k = 0 & \text{egyébként} \end{cases} \tag{2.1}$$

ahol τ_{ij} jelöli az i-edig és j-edig csúcs közötti feromon szintet, $\eta_{ij} = \frac{1}{d_{ij}}$ a két csúcs közötti él reciproka, α és β pedig paraméterek. Ha $\alpha = 0$ akkor a közelebbi városok kiválasztása kap nagyobb hangsúlyt, ha pedig $\beta = 0$, akkor a feromon szintek játszanak csak szerepet. $\mathcal{N}_i^k$ jelöli azon csúcsok halmazát, amelyek az adott hangya számára az i-edik csúcsból választhatóak. A gyorsabb konvergencia érdekében bevezethető az úgynevezett candidate list, amely minden csúcshoz tartalmazza a hozzá $k_{candidate}$ legközelebbi csúcsot. A candidate list hatása szintén a $\mathcal{N}_i^k$ halmazba kerülhet érvényesítésre.

A globális optimum megtalálása érdekében érdemes véletlen faktort is használni a csúcsok kiválasztása során, nemcsak a valószínűségi értékekre hagyatkozni. Ezért bevezetésre kerül $0 \leqslant q_0 \leqslant 1$ konstans, és minden csúcs választása előtt egy $q \in [0, 1]$ random szám. Ha $q_0 < q$ áll fenn, akkor a k-adik hangyához tartozó tabu list-be nem tartozó csúcsok közül véletlenszerűen kerül kiválasztásra a következő csúcs. Ellenkező esetben, a tabu list-be nem szereplő, de a candidate list-be szereplő csúcsok közül választhatunk a valószínűségi értékek alapján, ekkor tehát a tabu list és a candidate list alapján együttesen kerül meghatározásra a $\mathcal{N}_i^k$ halmaz. Ezt a legegyszerűbben úgy tehetjük meg, hogy minden csúcshoz generálunk egy véletlen számot, majd pedig a valószínűség értékeket ezzel súlyozzuk, és ennek maximumának helyét választjuk. Ha a tabu list-be már minden candidate list-be szereplő csúcs szerepel, akkor csak a tabu list alapján kerül meghatározásra a $\mathcal{N}_i^k$ halmaz.



2.1. ábra. A 2-OPT algoritmus szemléltetése. [15]

Az útvonalak megalkotása után, ha tehát már minden hangya minden csúcspontot meglátogatott, az egymással szomszédos csúcsok között 2-OPT algoritmust [26] alkalmazok. Tehát az útvonalon végig haladva két-két csúcsot megcserélek, és ha az így kapott útvonal rövidebb mint az eredeti akkor az útvonal a módosított módon kerül majd a to-

vábbi lépésekhez az ACO algoritmusban. Ellenkező esetben az eredeti tervezett csúcspont sorrend kerül alkalmazásra. Ennek számítási kapacitása $\mathcal{O}(N)$, N jelöli a csúcsok számát, azonban a konvergenciát jelentősen gyorsítja. A 2-OPT algoritmus alkalmas bizonyos önmagát metsző útvonalak kibogozására mint ahogy azt a 2.1 ábra mutatja, így csökkentve az útvonal hosszát. Fontos megjegyezni, hogy az ACO algoritmus önmagában is képes erre, több iteráció futtatása során, tehát a 2-OPT lokális keresés gyorsítja a konvergenciát, nem ad új megoldást. Jobb megoldást adna, ha minden csúcs felcserélését vizsgálnánk minden csúccsal, azonban ennek számítási kapacitása $\mathcal{O}(N^2)$ lenne ami jelentősen növelné a futási időt. A 2-OPT algoritmus önmagában is alkalmazható a TSP megoldására, ha az utóbb feltüntetett verzióját alkalmazzuk, azonban ennek nagy a számítási kapacitása, ugyanis a felcseréléseket addig kell ebben az esetben ciklikusan végrehajtani, amíg már nem keletkezik rövidebb útvonal.

A következő lépés az útvonal bejárásához szükséges idő kiszámítása, az út hosszának és az adatletöltés idejének figyelembevételével. Ennek során kihasználjuk az omnidirekcionális robot azon tulajdonságát, hogy pillanatszerűen képes irányváltoztatásra. Feltesszük továbbá, hogy az adatletöltést a csomópont alá helyezkedve álló helyzetben végzi, illetve, hogy két csomópont között a sebessége v konstans. Jelölje a k-adik hangya által készített útban a csomópontok sorrendjét Σ^k, Σ_i^k pedig az i-edik csomópontot a sorrendben. Ekkor a k-adik hangya által létrehozott útvonal adatgyűjtéshez szükséges T^k idő a következő módon számítható:

$$T^k = \sum_{i=1}^{n-1} \frac{d_{\Sigma_i^k, \Sigma_{i+1}^k}}{v} + \frac{d_{\Sigma_n^k, \Sigma_1^k}}{v} + 2\sum_{i=2}^{n} g_i \tag{2.2}$$

Az egyes hangyák által létrehozott útvonalak hosszának ismeretében a következő lépés a feromon frissítés, $\Delta\tau_{ij}$ meghatározása. Ez két részből tevődik össze, egyrészt frissítésre kerül az összes megalkotott úthossz alapján, másrészt globálisan a ciklusban létrehozott legrövidebb úthossz alapján:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L^k} + \frac{Q_{best}}{L_{best}} & \text{ha } k = best \\ \frac{Q}{L^k} & \text{egyébként,} \end{cases} \tag{2.3}$$

ahol

$$L^k = T^k\, v \tag{2.4}$$

jelöli a k-adik hangya által létrehozott útvonal bejárási idejét átszámítva úthosszba, L_{best} jelöli a ciklusban létrehozott leggyorsabb útvonal hosszát, Q és Q_{best} pedig súlyozó konstans paraméterek.

A feromon frissítése során figyelembe vesszük, hogy a régebbi feromon értékek az ciklusról ciklusra egyre kevésbé éreztetik hatásukat:

$$\tau_{ij}^k(t+1) = (1-\rho)\tau_{ij}^k(t) + \Delta\tau_{ij}^k \tag{2.5}$$

$$\tau_{ij}(t+1) = \sum_{k=1}^{m} \tau_{ij}^k(t+1) \tag{2.6}$$

ahol $(t+1)$ index jelöli a frissített feromon értéket, ρ pedig a felejtési tényező. A feromon frissítést minden ciklus elején hajtjuk végre. A hangya algoritmus lépéseit az utazóügynök megoldása során az Algoritmus 1 foglalja össze.

Algoritmus 1. – Hangya kolónia algoritmus alkalmazása utazóügynök probléma megoldására

1. d_{ij}, η_{ij}, τ_{ij} inicializálása, candidate list meghatározása
2. A ciklus maximális számaszor ismételve:

1. A feromon szintek frissítése: $\tau_{ij}^k(t+1) = (1-\rho)\tau_{ij}^k(t) + \Delta\tau_{ij}^k$, minden $k \in [1, m]$ hangyára elvégezve
2. A tabu list inicializálása, hangyák lehelyezése random kezdeti helyre
3. Amíg az összes N város be nem kerül a tabu list-be:

    Az összes m hangyára elvégezve:

    p_{ij}^k valószínűség kiszámítása, az aktuális i csúcsból a többi j csúcsba.

    q random szám generálása

    ha $q < q_0$

    ha van kiválasztatlan csúcs a candidate list-ben: csúcs választása a candidate list és tabu list alapján

    ha nincs kiválasztatlan csúcs a candidate list-ben: csúcs választása a tabu list alapján

    különben: következő csúcs random generálása a tabu list figyelembe vételével.

    tabu list frissítése a választott csúcs alapján

4. 2-OPT algoritmus futtatása a szomszédos csúcsok felcserélésével.
5. A ciklusban keletkezett legrövidebb út kiválasztása és mentése
6. $\Delta\tau_{ij}^k$ kiszámítása

3. A ciklusok legjobb útjaiból a legrövidebb kiválasztása

2.2. Multiágensű utazó ügynök probléma megoldása hangya kolónia algoritmus alkalmazásával

Mivel a robotok rendelkeznek fizikai kialakításuk miatt maximális sebességgel az adatgyűjtés során kritikus a megtervezett útvonal. Annak érdekében, hogy az adatkésleltetés minél kisebb legyen alkalmazható egyidejűleg több együttműködő robot, egy multiágensű rendszer. A feladat továbbra is az adat minél hamarabbi begyűjtése. A robotrendszer minden ágense a bázis csomópontból indul, együttesen az összes csomópontról letöltik az adatot, majd pedig a báziscsomópontra visszatérve feltöltik. Az elvégzendő feladat átkonvertálható Mutliple-TSP, azaz több kereskedőre kiterjesztett utazóügynök probléma megoldására Az MTSP problémának számos különböző esete és megoldási lehetősége ismert mint ahogy azt a [13] irodalom foglalja össze. Az MTSP problémát a 2.1 fejezethez hasonlóan hangya kolónia optimalizálás alkalmazásával oldom meg.

A több kereskedőre kiterjesztett Multiple-TSP probléma megoldására hangya kolónia optimalizációval számos lehetőség fellelhető a szakirodalomban [14] [15] [16] [17] [18]. A munkám során az alapötlet az, hogy egy útvonalat tervez minden hangya, amely $k-1$-szer tartalmazza a bázis csomópontot [14]. A költség függvényt az alúthosszak maximumának választottam meg minden hangya esetén (MinMax-MTSP). A feromon frissítés az alúthosszak összege alapján minden hangyára, illetve a legkisebb költségű hangyára a költség függvény alapján kerül meghatározásra. Az algoritmust az Algoritmus 2 foglalja össze.

Az algoritmus hasonló a már 2.1 fejezetben összefoglalthoz. Az első lépésben a d_{ij}, η_{ij}, τ_{ij} candidate list inicializálásán kívül meghatározásra kerül a két bázis csomópont között elhelyezkedő érzékelő csomópontok minimális és maximális száma is:

$$minN = \max\left\{\left\lfloor\frac{N}{0.75k}\right\rfloor, 3\right\} \tag{2.7}$$

$$maxN = \max\left\{\left\lfloor\frac{N+k-1-minN}{k-1}\right\rfloor, 3\right\} \tag{2.8}$$

A $maxN$ és $minN$ meghatározása során is fontos, hogy háromnál nem lehetnek kisebbek, hiszen akkor az algoritmus szakaszokat és pontokat is létre tudna hozni zárt alakzatok helyett. A másik fontos kritérium, hogy a maximális szám úgy kerüljön meghatározásra, hogyha az első $k-1$ alútvonal $maxN$ hosszú, az utolsó alútvonal még minimum $minN$ hosszúságú lehessen.

Az algoritmus 2.2 lépésében hangyák véletlenszerű lehelyezése helyett minden hangya a kezdő csomópontból indul. Erre a $maxN$ és $minN$ könnyebb kezelhetőségének érdekében van szükség, azonban ez ronthatja a konvergenciát ezért több ciklus futtatása javasolt. Az algoritmus 2.3 lépésében a következő választott csúcspontnál figyelembe kell venni a minimálisan és maximális csomópont számokat az alútvonalakban. Ha az utolsó bázis csomópont választás óta már minimum $minN$ számú város kiválasztásra került, akkor a bázis csomópont újra választható. Ekkor tehát a bázis csomópont látszólag kikerül a tabu list-ből. Ha az utolsó bázis csomópont után már kiválasztásra került $maxN$ számú csúcs akkor a bázis csomópont lesz a következő választott csúcs. Ekkor tehát a valószínűség alapján történő csúcsválasztás nem kerül végrehajtásra. A 2.4 lépésben k darab alútvonalra darabolom az útvonalat. A 2.5 lépésben az alútvonalak mindegyikére végrehajtom a 2-OPT algoritmust, majd az eredményül kapott utat úgy rotálom, hogy a bázis állomás legyen a kezdő csomópont. A hangya költségfüggvénye az alúthosszaknak a maximuma, tehát:

$$L^h_{cost} = \max_i l^h_i \tag{2.9}$$

Algoritmus 2. – Hangya kolónia algoritmus alkalmazása többszörös utazóügynök probléma megoldására (k-ACO-TSP)

1. d_{ij}, η_{ij}, τ_{ij} inicializálása, candidate list meghatározása, $minN$, $maxN$ kiszámítása

2. A meghatározott ciklus számszor ismételve:

1. A feromon szintek frissítése: $\tau_{ij}^h(t+1) = (1-\rho)\tau_{ij}^h(t) + \Delta\tau_{ij}^h$, minden $h \in [1, m]$ hangyára elvégezve
2. A tabu list inicializálása, hangyák lehelyezése a bázis csomópont kezdeti helyre
3. Amíg az összes N város és a bázis csomópont $k-1$-szer be nem kerül a tabu list-be:

   Az összes m hangyára elvégezve:

   Ha az utolsó bázis csomópont választás óta már minimum $minN$ számú város kiválasztásra került, akkor a bázis csomópont újra választható.

   Ha az utolsó bázis csomópont után már kiválasztásra került $maxN$ számú csúcs akkor a bázis csomópont lesz a következő választott csúcs. Különben:

   p_{ij}^k valószínűség kiszámítása, az aktuális i csúcsból a többi j csúcsba.

   q random szám generálása

   ha $q < q_0$

   ha van kiválasztatlan csúcs a candidate list-ben: csúcs választása a candidate list és tabu list alapján

   ha nincs kiválasztatlan csúcs a candidate list-ben: csúcs választása a tabu list alapján

   különben: következő csúcs random generálása a tabu list figyelembe vételével.

   tabu list frissítése a választott csúcs alapján
4. Az összesített út szétdarabolása k alútra, mindegyik a bázis csomópontnál kezdődjön.
5. 2-OPT algoritmus futtatása a szomszédos csúcsok felcserélésével az összes hangya összes alútvonalára, az algoritmus után a bázis csomópont visszaállítása kezdőcsomópontra
6. Az alútvonalak hosszának meghatározása, minden hangya költségfüggvényének kiszámítása: A leghosszabb alútvonal hossz.
7. A ciklusban keletkezett legkisebb költségű út kiválasztása és mentése
8. $\Delta\tau_{ij}^h$ kiszámítása, az alúthosszak összege alapján minden hangyára és a költség függvény alapján a legjobb (legkisebb költségű) hangyára.

3. A ciklusok legjobb útjaiból a legrövidebb kiválasztása

ahol l_i^h jelöli a h-adik hangya i-edik alútvonalának a hosszát. Jelölje L_{sum}^h a h-adik hangya alútvonalainak az összegét:

$$L_{sum}^h = \sum_{i=1}^{k} l_i^h. \tag{2.10}$$

A feromon frissítés a h hangyára következő formában írható le, ha az (i, j) élt tartalmazza a h hangya útvonala.

$$\Delta\tau_{ij}^{h} = \begin{cases} \frac{Q}{L_{sum}^{h}} + \frac{Q_{best}}{L_{cost}^{h}} & \text{ha } h = best \\ \frac{Q}{L_{sum}^{h}} & \text{egyébként.} \end{cases} \tag{2.11}$$

A h hangya teljes feromon frissítése a ρ felejtési tényező figyelembevételével

$$\tau_{ij}^{h}(t+1) = (1-\rho)\tau_{ij}^{h}(t) + \Delta\tau_{ij}^{h} \tag{2.12}$$

Az összes ciklus elvégzése után a legkisebb költségű útvonal kerül kiválasztásra a probléma megoldására.

Megjegyzés. A hangyakolónia algoritmus a véletlen szám generálás miatt nem ad determinisztikus megoldást. Ezért többszöri futtatás során más-más eredmény lehetséges. Ez a hatás maximális ciklusszám növelésével csökken, mivel az optimális megoldás felé konvergál. ▪

3. fejezet

Útvonal tervezés prioritásos csomópontok között

Egy érzékelő terület bejárása során előfordulhat, hogy az egyes csomópontok meglátogatása sürgetőbb mint a többié. Például bajba jutott sebesült emberek felkutatása során a súlyosabb sérültek hamarabb ellátásra szorulnak amit már a mentőegység útközben is elkezdhet. Természetesen ekkor is fontos a lehető leggyorsabb bejárás, hogy a lehető leghamarabb kerüljenek ellátásra a betegek. Ezért ebben a fejezetben prioritással rendelkező csomópontok közötti útvonal megtervezésére kerül sor. A robotok adott kezdő pontból indulnak és egy adott végpontba érkeznek meg, miközben az összes csomópontot meglátogatják figyelembe véve a csomópontok prioritását és minimalizálják a bejárás idejét. Kiinduló pont lehet minden robot számára azonos, például egy mentőakció során a bázis állomásról indul minden alakulat. Azonban előfordulhat olyan eset is amikor véletlenszerű helyről indulnak ki, ilyen a bajba jutott egymás segítségére igyekvő síelők esete. A megoldásom során feltételezem, hogy egy közös végpontba érkeznek meg a robotok, de a különböző végpontokba való megoldás sem tér el sokban a megoldásomtól. A következőkben ismertetem az általános, különböző csomópontokból induló és azonos végpontba beérkező robotok adatgyűjtésére elkészített algoritmust, amelynek egy speciális esete a közös kiinduló pontból készített útvonal.

A prioritásos útvonaltervezés visszavezethető egy legrövidebb költségű Hamilton út megtalálására. Ez az utazóügynök problémának egy általánosított esete, tehát szintén NP-nehéz, ezért megoldásához az előzőekhez hasonlóan hangya kolónia algoritmust használok. Ennek fő lépései megegyeznek az Algoritmus 1 felvázoltakkal. A prioritásos útvonaltervezés lépéseit az Algoritmus 3 foglalja össze.

Adott a kezdő- (Start) és végpont(ok) (End) illetve az n darab érzékelő csomópont helye, az ezeken tárolt g_i adat és a P_i prioritásuk. A legfontosabb csúcs 1-es prioritású, a prioritással nem rendelkező csomópontok pedig n'- kezdő és végpontokkal együttesen vett csomópontszám- prioritást kapnak automatikusan. Ismert a robot v haladási sebessége és a csomópontok és a robot közötti r adat átviteli sebesség. Első lépésben meghatározásra kerülnek a csomópontok közötti d_{ij} távolságok, illetve az $\eta_{ij} = 1/d_{ij}$ és a szomszédsági listák is. A szomszédsági, candidate, listákban a kezdő pontok nem szerepelhetnek szomszédokként, azonban ezek is rendelkeznek candidate listával. A végpont szerepelhet a szomszédsági listákban, azonban nincsen saját listája. Ilyen módon a candidate list antiszimmetrikus. Inicializálásra kerül a τ_{ij} feromon szint a gráf élein és az egyidejűleg útvonalat építő m hangyák száma is. Az adott P_i prioritások felhasználásával létrehozom a P_i^F prioritásfiltert

Algoritmus 3. – Hangya kolónia algoritmus alkalmazása prioritásos csomópontok közötti útvonaltervezésre

1. d_{ij}, η_{ij}, τ_{ij} inicializálása, candidate list és P_i^F prioritásfilter meghatározása

2. A meghatározott ciklus számszor ismételve:

1. A feromon szintek frissítése: $\tau_{ij}^h(t+1) = (1-\rho)\tau_{ij}^h(t) + \Delta\tau_{ij}^h$, minden $h \in [1, m]$ hangyára elvégezve
2. A tabu list inicializálása, hangyák lehelyezése a Start pontokba
3. Amíg az összes N város és a végpont be nem kerül a tabu list-be:

   Az összes $h \in [1, m]$ hangyacsapatra és ezek $s \in [1, k]$ hangyájára elvégezve:

   Ha s hangya útvonala tartalmazza End csomópontot ugrás a következő hangyára

   Ha s útvonala tartalmaz már érzékelő csomópontot az End csúcs választható

   Ha s összes csapattársának útvonala tartalmazza az End pontot és még van bejáratlan érzékelő csomópont, akkor s nem választhatja End pontot.

   p_{ij}^k valószínűség kiszámítása (3.2) egyenlet alapján, az aktuális i csúcsból a többi j csúcsba.

   q random szám generálása

   ha $q < q_0$

   ha van kiválasztatlan csúcs a candidate list-ben: csúcs választása a candidate list és tabu list alapján

   ha nincs kiválasztatlan csúcs a candidate list-ben: csúcs választása a tabu list alapján

   különben: következő csúcs random generálása a tabu list figyelembe vételével.

   tabu list frissítése a választott csúcs alapján
4. Az részútvonalak hosszának és költségének meghatározása, minden hangya költségfüggvényének kiszámítása: a legnagyobb részútvonal költség.
5. A ciklusban keletkezett legkisebb költségű út kiválasztása és mentése
6. $\Delta\tau_{ij}^h$ kiszámítása az (3.5) egyenlet alapján

3. A ciklusok legjobb útjaiból a legkisebb költségű kiválasztása

az (3.1) egyenlet alapján:

$$P_i^F = \frac{\max\limits_{j \in n'} P_j + 1 - P_i}{\max\limits_{j \in n'} P_j}. \tag{3.1}$$

Ilyen módon, minél fontosabb egy csúcs, annál nagyobb érték tartozik hozzá a prioritásfilterben.

Az inicializálás után kezdetét veszi a fő ciklus meghatározott számszor ismételve, amelynek három fő szakasza van: inicializálás, útvonal építés, feromon frissítés és útvonalmentés. Ennek első lépése a feromon szint frissítése az előző ciklus eredményei alapján,

majd pedig a tabu lista inicializálása. A hangyák is lehelyezésre kerülnek az útvonaluk kezdő pontjaira. Jelölje k a bejáráshoz használt ágensek számát. Ekkor úgy tekinthetünk minden egyes hangyára mint egy k hangyából álló csapatra. Tehát az inicializálás során $k\,m$ darab hangya kerül lehelyezésre úgy, hogy azonos csapatba tartozó hangyáknak az összes Start pontot le kell fedniük. Az egy Start pontos esetben minden hangya ugyanonnan indul.

Ezt követően kezdődhet az útvonaltervezés. Minden hangya csapat addig épít útvonalat, amíg minden csomópontot meg nem látogattak, tehát amíg a tabu listában nem szerepel az összes csomópont. A hangya csapat tagjainak nem muszáj azonos számú csomópontot meglátogatniuk, azonban a kezdő és végponton kívül minimum egy csomópont meglátogatandó. Ezt követően már minden hangyának adott a lehetőség, hogy a végpontot válassza következőnek, azonban addig nem érhet véget az útvonaltervezés a hangya csapat számára amíg van nem meglátogatott csomópont. Tehát, ha a hangya csapat $k-1$ darab hangyája már választotta az End pontot, akkor az utolsó hangya csak akkor fejezheti be az útvonalát, ha a maradék érzékelő csomópontokat meglátogatta. A hangya csapat tagjai útvonaltervezésének lépései megegyeznek az Algoritmus 1-ben felvázoltakkal. Azonban itt a p_{ij}^h valószínűség kiszámítása során súlyozásra kerül a P_j^F értékkel:

$$p_{ij}^h = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum\limits_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} P_j^F & \text{ha } j \in \mathcal{N}_i^k \\ p_{ij}^h = 0 & \text{egyébként} \end{cases} \tag{3.2}$$

A következő lépés az egyes készített rész utak, tehát a hangya csapatok egyes tagjai által készített utak, hosszának és költségének meghatározása. A h-adik hangya csapat s-edik tagja által létrehozott út L_s^h hossza az (2.2) és (2.4) alapján kerül meghatározásra. A költség két részből tevődik össze, egyrészt az L_s^h hosszból, másrészt a M_s^h prioritási költségből. A prioritási költség az i-edik csúcsnál a Σ_s^h permutációban, arányos az adott csúcs eléréséig eltelt idő és a csúcs prioritásának szorzatával. Az s hangya J_s^h költsége tehát:

$$J_s^h = L_s^h + \frac{\sum\limits_{i \in \Sigma_s^h} L_s^h(i) P^F_{\Sigma_s^h(i)}}{L_s^h} = L_s^h + \sum_{i \in \Sigma_s^h} D_s^h(i) = L_s^h + D_s^h, \tag{3.3}$$

ahol $L_s^h(i)$ a Σ_s^h permutációban az i-edik csúcsig kiszámított adatgyűjtési úthossz, $\Sigma_s^h(i)$ pedig a h-adik hangyacsapat s-edik részútvonalának i-edik csúcsa. A h-adik hangya költsége a hangyacsapat részköltségeinek maximuma:

$$J^h = \max_s J_s^h \tag{3.4}$$

A feromon frissitést minden olyan élre el kell végezni, amelyet bármelyik hangya csapat bármelyik hangyája választott. A feromon frissítés három tagból áll: minden alútvonalból számított prioritás költség a j-edik csúcsig minden s hangyára, a hangyacsapat költsége és a legkisebb költség a ciklusban (best). Ha h hangya bármelyik s hangyája használja az útvonala során az (i, j) élt, akkor a feromon frissítés a következőképpen zajlik:

$$\Delta\tau_{ij}^h = \begin{cases} \frac{Q_1}{J^h} + \frac{Q_2}{D_s^h(j)} + \Delta\tau_{ij}^{gb} & \text{ha h a legjobb hangyacsapat} \\ \frac{Q_1}{J^h} + \frac{Q_2}{D_s^h(j)} & \text{különben} \end{cases} \tag{3.5}$$

ahol $\Delta\tau_{ij}^{gb}$ a ciklusban keletkezett legjobb hangya útvonala utáni feromon frissítés:

$$\Delta\tau_{ij}^{gb} = \frac{Q_{best}}{J_{best}}, \tag{3.6}$$

itt csak azon (i, j) élek kerülnek frissítésre amelyek szerepelnek a legjobb hangya csapat bármely s útvonalában. Tehát a h hangyacsapat feromon frissítése a ρ felejtési tényező figyelembevételével:

$$\tau_{ij}^{h}(t+1) = (1-\rho)\tau_{ij}^{h}(t) + \Delta\tau_{ij}^{h} \tag{3.7}$$

A ciklus végén mentésre kerül a legkisebb költségű, legjobb útvonal. Az adott számú ciklus elvégzése után a legjobb útvonalak közül a legkisebb költségű kerül kiválasztásra megoldásként.

4. fejezet

Szimulációs eredmények

A szimulációs mező egy $200m \times 200m$ nagyságú terület, amelyen periodikus esetben 39, prioritásos esetben pedig 40 érzékelő csomópont helyezkedik el. A robot haladási sebessége a csomópontok között $v = 4m/s$. Minden érzékelő csomóponton $g_i = 0.5MB$ adat van tárolva, az adatátviteli sebesség $r_i = 250kB/s$.

A numerikus stabilitás miatt a csomópontokat $[0, 1] \times [0, 1]$ intervallumra normálom. A távolságmátrix, d_{ij} értékek, meghatározása során euklideszi távolságot számítok a már normált csomópontok között. A candidate list meghatározása során minden csomóponthoz a hozzá, a csomópontok számának 20% legközelebbi csomópontot rendeljük. Tehát minden csomóponthoz $n \cdot 0.2$ számú csomópont tartozik. A hangya kolónia algoritmusnak számos paramétere van: α, β, ρ, q_0, Q és Q_{best}. Számos irodalom foglalkozik ezen paraméterek optimális megválasztásával [23] [24] [27] [28]. A munkám során én a klasszikus [23] paraméterekkel implementáltam az algoritmust: $\alpha = 1$, $\beta = 2$, $\rho = 0.1$, $q_0 = 0.9$ és $Q = Q_1 = Q_{best} = 1$. A Q_2 értéket tapasztalati úton $Q_2 = Q_1/5$ értékre választottam. Ez biztosítja az egyensúlyt a teljes útvonal és a részútvonal szerinti feromon frissítés között. Maximális ciklusszámnak tapasztalati úton a periodikus és prioritásos útvonaltervezéshez rendre 1000 és 5000 értéket választottam, így a futási idő értéke megközelítőleg 90sec periodikus és 210sec a prioritásos esetben.

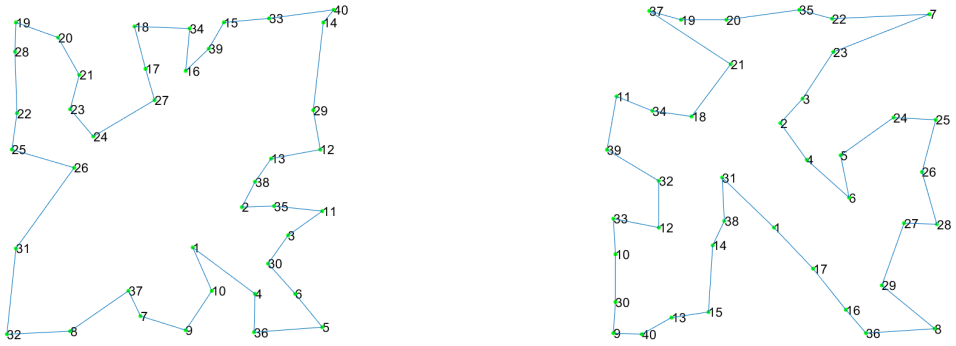
Fontos még az alkalmazott hangyák m száma. Ha túl kevés hangya kerül alkalmazásra egy-egy ciklusban, akkor nem lesz eléggé determinisztikus a bejárás az aktuális feromon értékek alapján. Azonban ha túl sok hangya épít útvonalat egy ciklus során, a nagy számítási kapacitás mellett a feromon frissítésnél túl nagy hangsúlyt fognak kapni az adott ciklusban épített utak, az előző ciklusokban meghatározott feromon értékekhez képest. Túl sok hangya használatakor szintén előfordulhat, hogy több ugyanolyan útvonal keletkezik, amelyek a feromon frissítésnél többszörösen fognak egy-egy élt frissíteni így torzítva a konvergenciát. A szimulációk során én $m = n/2$ számú hangyát, illetve prioritásos esetben hangyacsapatot használok.

A fejezetben először a periodikus útvonaltervezés eredményei, majd pedig a prioritásos útvonaltervezés eredményei kerülnek bemutatásra.

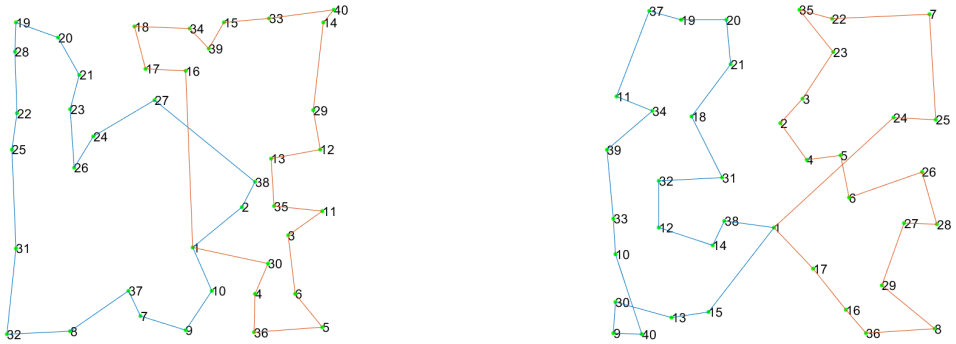
4.1. Periodikus útvonaltervezés eredményei

A periodikus útvonal tervezést hat különböző érzékelő mezőn, $k \in [1, 8]$ robotra hajtottam végre. A bázis csomópont minden esetben a Node-1, tehát az egyes sorszámú pont. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárását $k \in [1, 8]$ számú robotokkal a 4.1-4.8 ábrák mutatják.

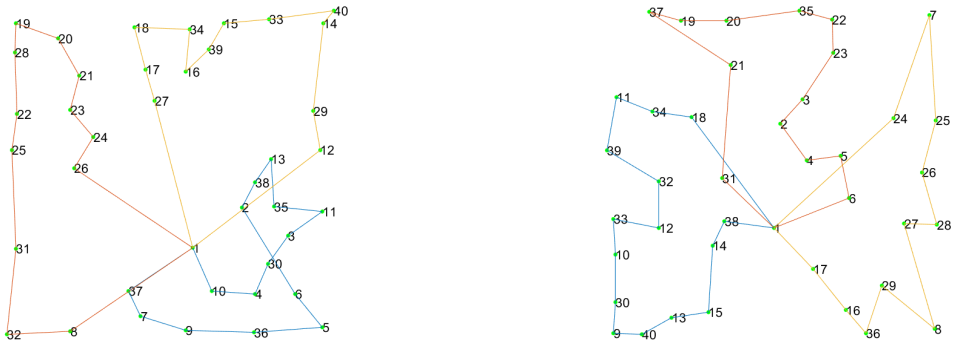
Az alútvonalak hossza a F.1 fejezetben kerültek feltüntetésre. A 4.1 ábrán jól megfigyelhető az érzékelő csomópontok elhelyezkedése. A Sensing Field 2 mezőn a csomópontok



4.1. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 1$ ágenst alkalmazva.

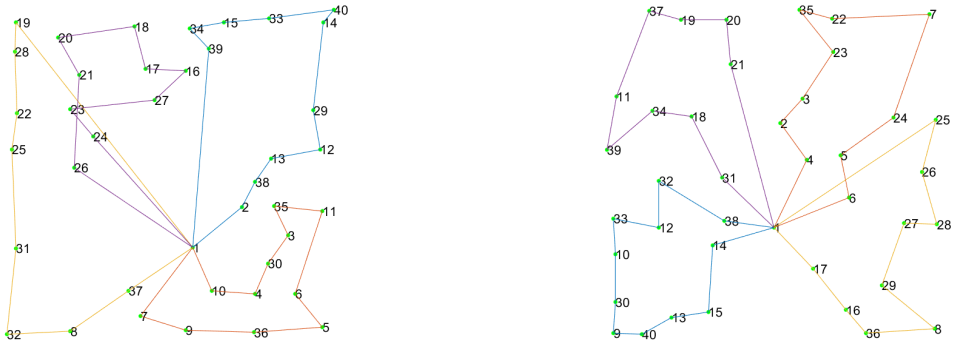


4.2. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 2$ ágenst alkalmazva.

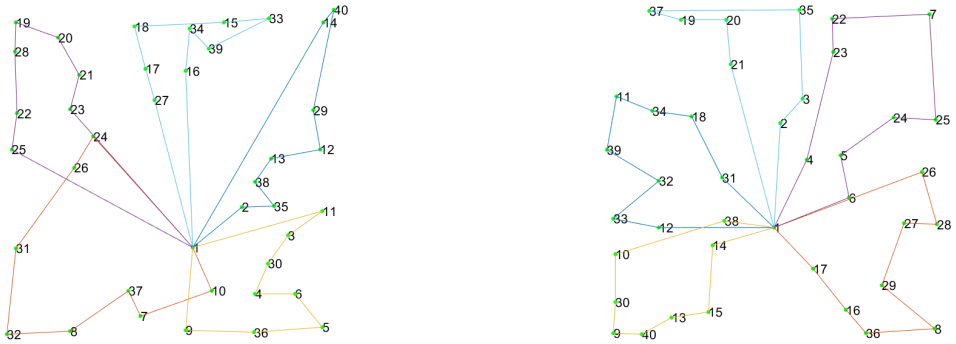


4.3. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 3$ ágenst alkalmazva.

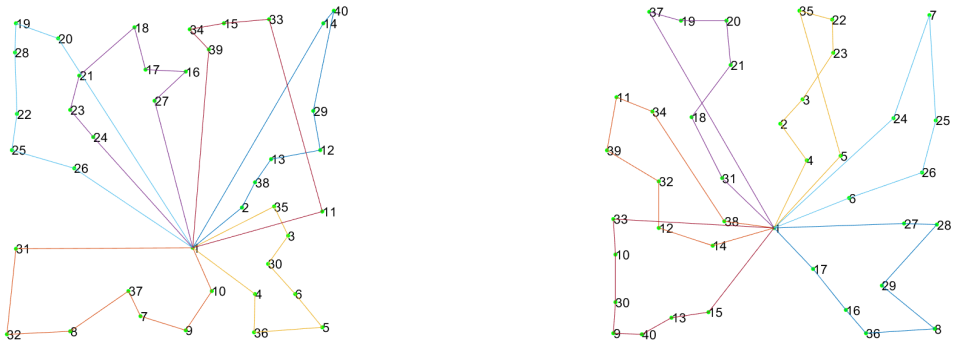
megközelítőleg két csoportra osztva helyezkednek el, az egyik átlót kihagyva, míg a Sensing Field 4 egyenletesebb eloszlást mutat. A Sensing Field 2 bázis csomópontja a jobb alsó sarokhoz közelebb helyezkedik el, aminek hatása több ágenses megoldásokban is jelentkezik. A 4.2 ábrán a Sensing Field 2 mező útvonalai egymást metszik, tehát valószínűleg rövidebb útvonal is tervezhető lett volna, ha a Node-2 és Node-38 a második, pirossal jelölt alútvonalhoz tartozna. A Sensing Field 4 mezőn egy pillangó szerű ábrát láthatunk,



4.4. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 4$ ágenst alkalmazva.

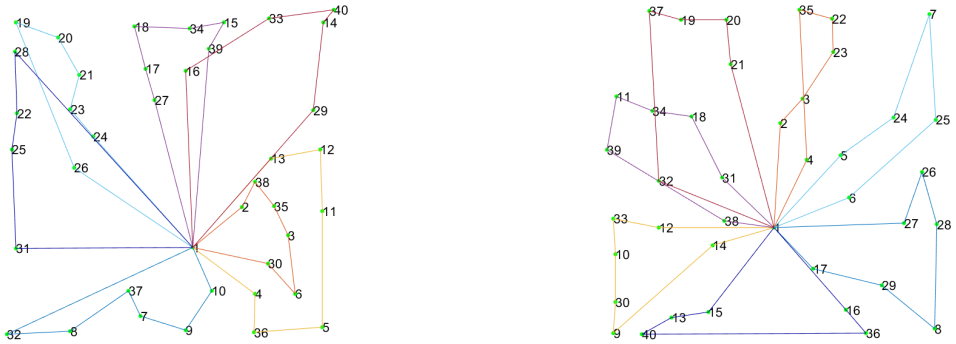


4.5. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 5$ ágenst alkalmazva.

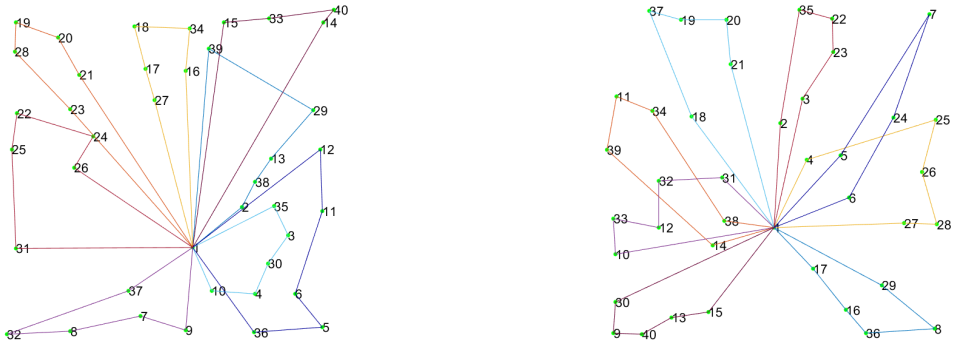


4.6. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 6$ ágenst alkalmazva.

azonban itt az útvonalak önmagukat metszik, amelyek szintén a nem optimális eredményt mutatják. Ezek a ciklusszám növelésével kiküszöbölhetők. A 4.3 ábrán az első, kék színnel jelölt alútvonal önmagát metszi, azonban ez a bejárási időt lényegesen nem befolyásolja, mivel a pirossal jelölt útvonal megközelítőleg azonos hosszúságú ($t_1 = 181,6s$, $t_2 = 180,9s$, $t_3 = 172,2s$). A 4.4 ábrán a második, pirossal jelölt útvonal lényegesen rövidebb, a bázis csomópont elhelyezkedése miatt ($t_1 = 150,7s$, $t_2 = 143,6s$, $t_3 = 157,1s$, $152,4s$). A lila



4.7. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 7$ ágenst alkalmazva.

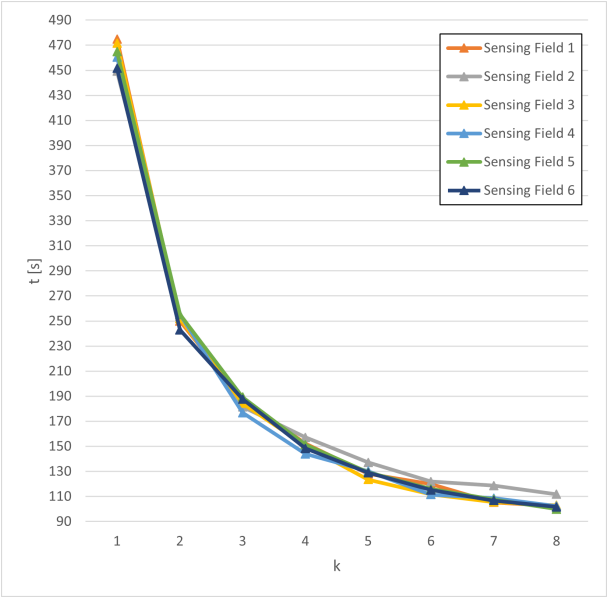


4.8. ábra. A Sensing Field 2 és Sensing Field 4 érzékelő mezők bejárása $k = 8$ ágenst alkalmazva.

útvonal önmagát metszi, de ez nem befolyásolja a csomópontok bejárási idejét, hiszen a sárga útvonal a leghosszabb. Az 1. és 2. videón (7 fejezet) látható a Sensing Field 2 és Sensing Field 4 mezőről való adatgyűjtés $k = 4$ robotot alkalmazva. Az 1. videón megfigyelhető, hogy a pirossal jelölt robot hamarabb befejezi az adatgyűjtést, így már akkor elkezdi a következő periódust amikor még társai az adatokat töltik fel a báziscsomópontra. A 2. videón a kékkel jelölt robot a leggyorsabb, a többi robot közel azonos idő alatt végzi el az adatgyűjtést. A 4.5 és 4.6 ábrákon a Sensing Field 2 bejárása során a sárgával jelölt útvonal lényegesen rövidebb a báziscsomópont elhelyezkedése miatt. A 4.5 ábrán a Sensing Field 2 mezőn a piros és lila színnel jelölt útvonalak metszik egymást, tehát csökkenthető lenne a bejárási hosszuk, de ez szintén nem befolyásolja a bejárási időt, mivel a világos kékkel jelölt útvonal a leghosszabb. Az önmagukat és egymást metsző útvonalak a hangya algoritmus során azért nem kerülnek optimalizálásra, mert a költségfüggvény csak a leghosszabb alútvonal hosszát figyeli, azt igyekszik csökkenteni. A Sensing Field 4 mezőn a bázis csomópont a mező közepéhez közel helyezkedik el, ezért egyenletesebb bejárás biztosított mint a Sensing Field 2 esetében. A tervezett alútvonalak virágszirmokhoz hasonlóan helyezkednek el, azonban itt is megjelennek egymást metsző útvonalak, 4.6-4.8 ábrákon.

A szimulációk értékeléséhez mérőszámokat használtam a [25] alapján. Először is az ágensek bejárási idejét vizsgálom, ezek maximuma adja a teljes adatgyűjtési időt. A hat különböző mezőn végzett $k \in [1, 8]$ ágensű szimulációkban a teljes adatgyűjtési időt a 4.9 ábra mutatja. A robotszám növekedésével a bejárási idő lényegesen csökken, 8 ágenst alkalmazva csaknem a negyedére az egy ágenst alkalmazó esethez képest. Azonban, a

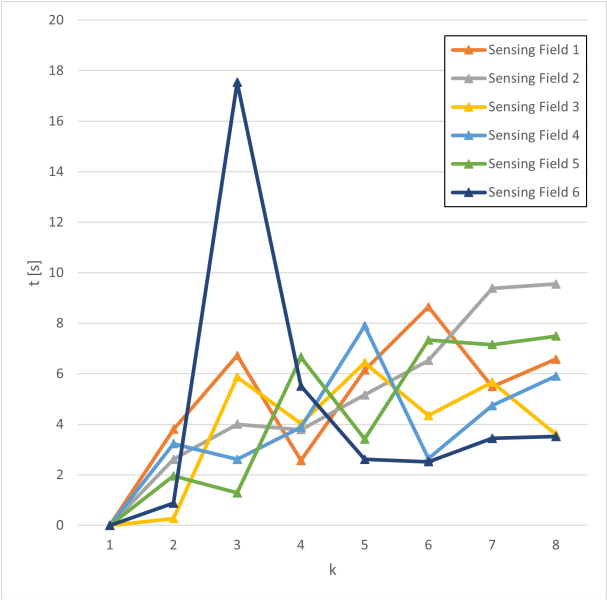
bejárási idő csökkenésének mértéke csökken az alkalmazott robotok számával, tehát minél több robot kerül alkalmazásra, annál kevésbé mérvadó a bejárási idő különbsége. $k = 4$ esetben megközelítőleg harmadára $k = 8$ esetben pedig negyedére csökkenthető a bejárási idő a vizsgált érzékelő mezők mindegyikén.



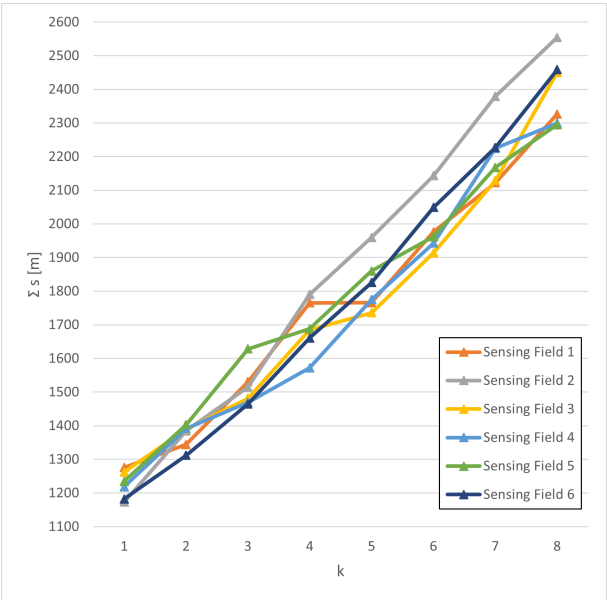
4.9. ábra. A hat különböző érzékelő mező bejárási ideje $k \in [1, 8]$ ágenst alkalmazva

A következő mérőszám az alútvonalakon történő adatgyűjtési idők átlagtól való átlagos eltérése, amely a csomópontok egyenlőtlen bejárási gyakoriságát mutatja. A hat különböző mezőn végzett $k \in [1, 8]$ ágensű szimulációkban az adatgyűjtési idők átlagtól való átlagos eltérését a 4.10 ábra mutatja. A növekvő robotszámmal nem monoton, de mégis növekvő tendenciát mutat. Tehát az ágensek számát növelve a csomópontok bejárási gyakoriságának különbsége nő, így hosszútávon egyenlőtlen adatgyűjtést eredményezve.

Az utolsó mérőszám a robotok összegzett úthossza, amelyet egy teljes adatgyűjtési ciklusban meg kell tenniük. Ez a robotok energia fogyasztásával arányos. A hat különböző mezőn végzett $k \in [1, 8]$ ágensű szimulációkban az összegzett úthosszat a 4.11 ábra mutatja. A robotszám növekedésével természetesen nő az összesen megtett úthossz, közel lineáris módon.



4.10. ábra. A hat különböző érzékelő mező adatgyűjtési idejének átlagtól való átlagos eltérése $k \in [1, 8]$ ágenst alkalmazva



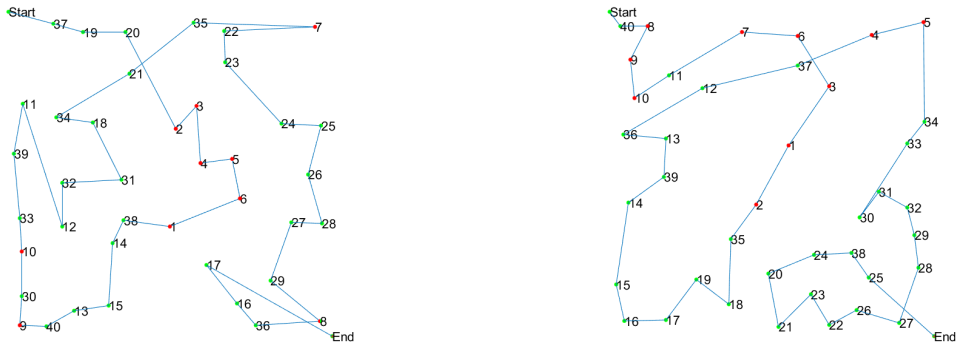
4.11. ábra. A hat különböző érzékelő mező összegzett úthossza $k \in [1, 8]$ ágenst alkalmazva

4.2. Prioritásos csomópontok közötti útvonaltervezés

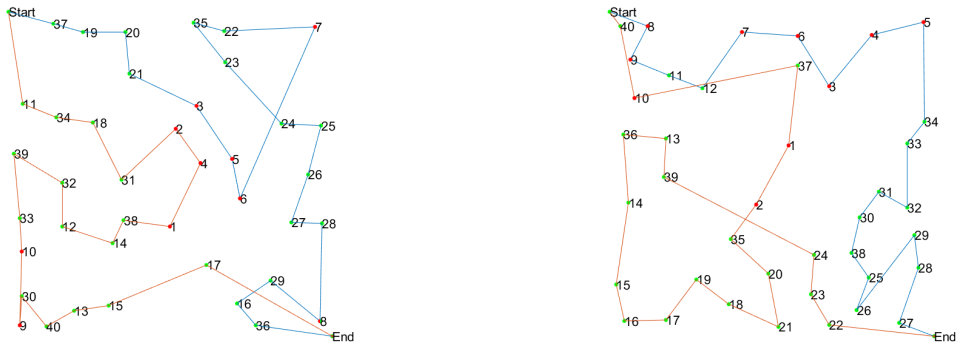
A prioritásos útvonaltervezést hat különböző mezőn, $k \in [1, 6]$ robotra hajtottam végre. Az első tíz sorszámú csomópont rendelkezik azonosan magas prioritással, míg a többi csomópont nem rendelkezik prioritással, azaz azonosan alacsony prioritással rendelkezik. Két féle esetet vizsgálok: az első esetben minden ágens az érzékelő mező bal felső sarkából indul és a jobb alsó sarokba kell megérkezniük, mint egy tipikus mentőexpedíció során. Míg a második esetben az ágensek véletlenszerű helyről indulnak és a mező közepére kell beérkezniük, ilyen például egy osztálykirándulás során a gyerekek begyűjtése.

4.2.1. Egy kezdő pontból induló ágensek esete

A alútvonalak bejárási idejét a F.2.1 fejezetben kerül feltüntetésre. A Sensing Field 4 és Sensing Field 6 mezőn történő adatgyűjtést a 4.12-4.17 ábrák mutatják. A 4.12 ábrán



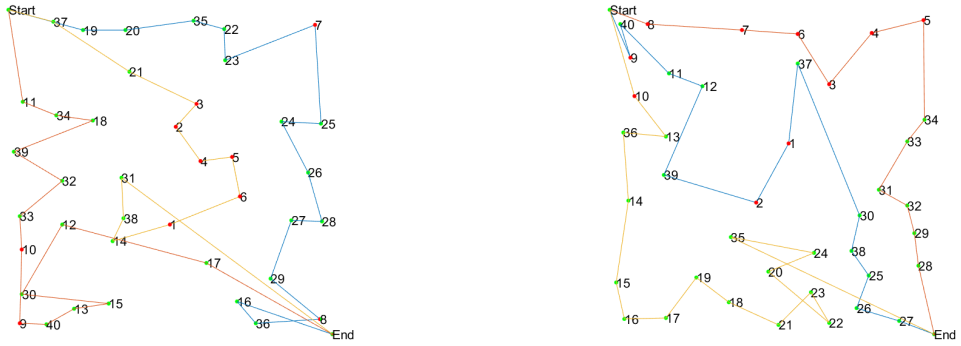
4.12. ábra. A Sensing Field 4 és Sensing Field 6 érzékelő mezők bejárása $k = 1$ ágenst alkalmazva.



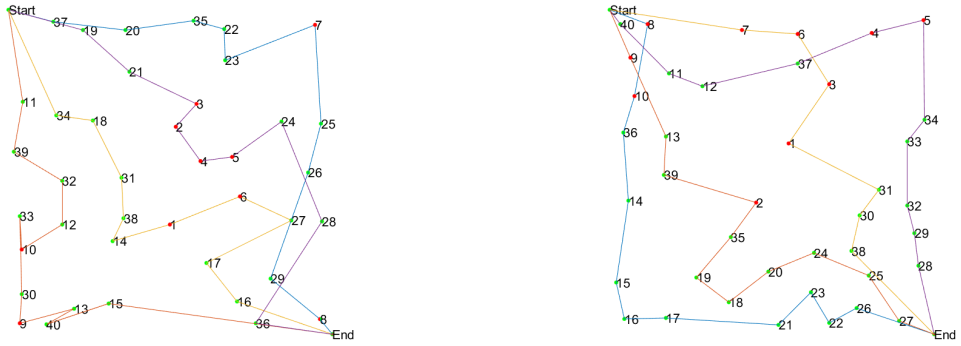
4.13. ábra. A Sensing Field 4 és Sensing Field 6 érzékelő mezők bejárása $k = 2$ ágenst alkalmazva.

megfigyelhető, hogy a Sensing Field 4 mezőn a prioritásos, pirossal jelölt (Node1-Node10) csomópontok közül Node-8 az End ponthoz közel helyezkedik el, míg a Sensing Field 6 mezőn a prioritásos csomópontok a Start ponthoz helyezkednek el közelebb. Ez jelentősen befolyásolja a tervezett útvonalat.

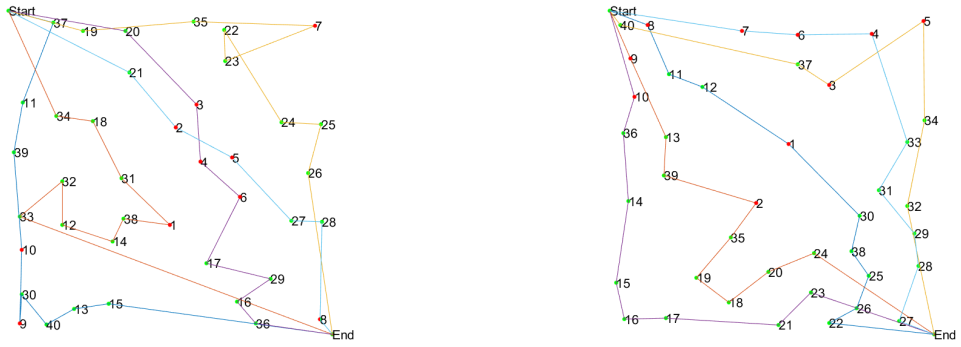
A 4.18 ábrán látható az egyes érzékelő mezőkön az adatgyűjtés ideje, tehát az alútvonalak maximális bejárási ideje, és az összes prioritásos csomópont eléréséhez szükséges idő $k \in [1, 6]$ ágens esetén. Leolvasható, hogy a Sensing Field 6 elrendezésének köszönhetően



4.14. ábra. A Sensing Field 4 és Sensing Field 6 érzékelő mezők bejárása $k = 3$ ágenst alkalmazva.



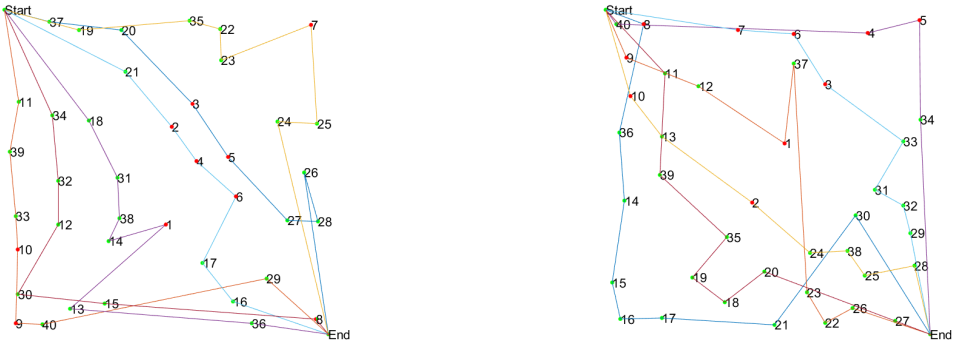
4.15. ábra. A Sensing Field 4 és Sensing Field 6 érzékelő mezők bejárása $k = 4$ ágenst alkalmazva.



4.16. ábra. A Sensing Field 4 és Sensing Field 6 érzékelő mezők bejárása $k = 5$ ágenst alkalmazva.

$k \in [2, 6]$ esetére kevesebb mint a teljes bejárási idő fele alatt a prioritásos csomópontok mindegyike már meglátogatásra került. Azonban az is jól látható, hogy ez az arány a Sensing Field 4 esetében lényegesen rosszabb, többek között a Node-8 csomópont szerencsétlen elhelyezkedése miatt.

A prioritással rendelkező csomópontok közötti útvonaltervezésnek fontos tulajdonsága, hogy a prioritásos csomópontokat milyen gyorsan sikerül meglátogatni. Ennek egy



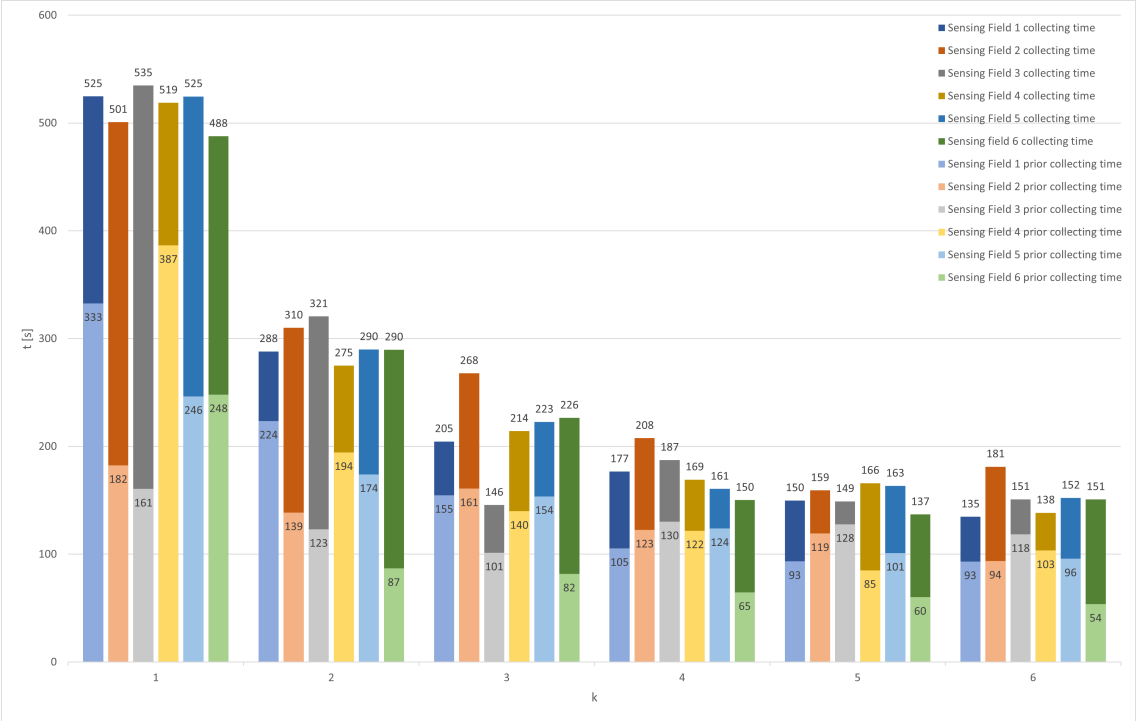
4.17. ábra. A Sensing Field 4 és Sensing Field 6 érzékelő mezők bejárása $k = 6$ ágenst alkalmazva.

mutatója, hogy a teljes bejárási idő feléig, hány prioritással rendelkező csomópont került meglátogatásra. A hat különböző érzékelő mezőn $k \in [1, 6]$ robotra ezt a 4.1 táblázat mutatja.

k	1	2	3	4	5	6
Sensing Field 1	9	9	9	9	9	9
Sensing Field 2	10	10	9	9	5	9
Sensing Field 3	10	10	5	5	1	4
Sensing Field 4	8	7	9	9	9	9
Sensing Field 5	10	9	9	7	7	7
Sensing Field 6	9	10	10	10	10	10

4.1. táblázat. A teljes bejárási idő feléig meglátogatott prioritásos csomópontok száma (10 prioritással rendelkező csomópontból)

A 4.12 ábrán önmagukat többszörösen metsző útvonalakat láthatunk, amely a prioritásos csomópontok minél előbbi meglátogatása miatt fordul elő főként. A Sensing Field 6 bejárása során az első 10 meglátogatott csomópontból 8 magas prioritással rendelkező csomópont. Az útvonal végén azonban prioritásos csomópontokat nem tartalmazó hurok alakult ki, ami növeli a bejárási időt. Ez a ciklusszám növelésével megszüntethető. A Sensing Field 4 bejárása során, az első 10 meglátogatott csomópontból már csak 6 magas prioritású, és az útvonal végén a Node-8 csomópont miatt hurok alakul ki. A Node-7 prioritásos csomópont meglátogatása során pedig kiugrás látható. A 4.13 ábrán az útvonalak sokkal kisimultabbak, csak a prioritásos csomópontok hamarabbi meglátogatása miatt tartalmaz hurkokat. A Sensing Field 4 mezőn hét, a Sensing Field 6 esetében, pedig mindegyik csomópont meglátogatásra kerül a teljes bejárási idő felének eltelte előtt a 4.1 táblázat szerint. $k \in [3, 6]$ esetén a Sensing Field 4 mezőn a Node-8 csomóponton kívül az összes, 9, a Sensing Field 6 mezőn pedig az összes, 10, csomópont meglátogatásra kerül a bejárási idő fele előtt. A 4.14 ábrán látható, hogy a Sensing Field 4 esetében az útvonalak csak akkor metszik önmagukat, amikor a prioritással rendelkező csomópontok hamarabbi meglátogatása a cél. Azonban a Sensing Field 6 esetén a sárga színnel jelölt útvonal nem prioritásos csomópontok között önmagát metsző utat eredményez, ami jelentősen növeli az út hosszát, és a bejárási időt is, mivel így ez lesz a leghosszabb útvonal. Mivel ez az út csak az elején tartalmaz prioritásos csomópontot, Node-10, a hangya algoritmusban használt minimalizálandó költsége kicsi a másik két alútvonalhoz képest. Ezért nem került kisimításra az algoritmus során. A $k = 4$ és $k = 5$ ágenst használó megoldás a 4.15 és

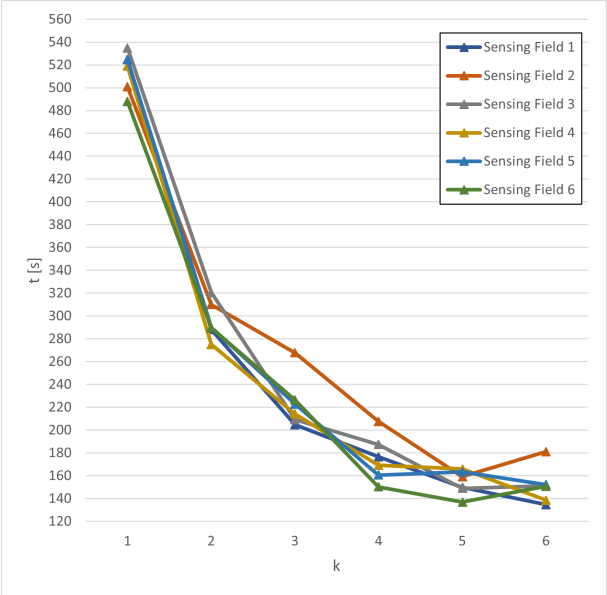


4.18. ábra. A hat különböző érzékelő mező teljes bejárási és az összes prioritásos csomópont meglátogatásához szükséges idő $k \in [1, 6]$ ágenst alkalmazva.

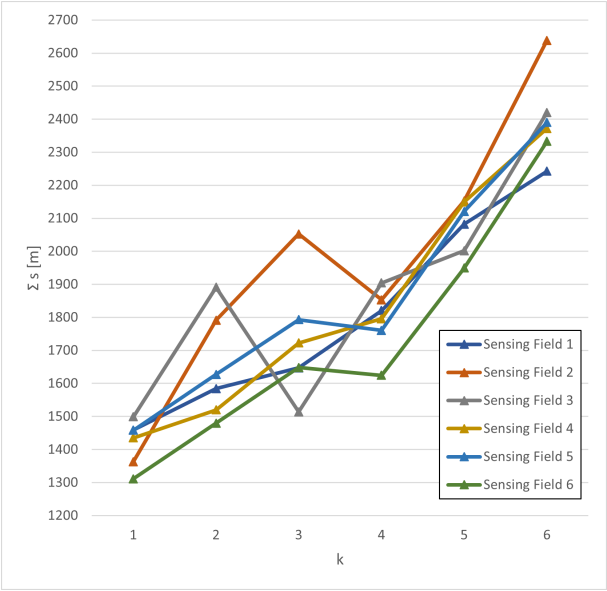
4.16 ábrákon láthatóak. A különböző ágensek útvonalai ebben az esetben jól szegregáltak, simák, önmagukat nem metszik. A 3. és 4. videón (7 fejezet) a Sensing Field 4 és Sensing Field 6 érzékelő mezőről való adatgyűjtést mutatja be $k = 4$ számára. Ezeken jól látható, hogy a Sensing Field 4 mezőn sokkal később kerül meglátogatásra az összes prioritással rendelkező csomópont, mint a Sensing Field 6 esetén a csomópontok elhelyezkedésének következtében. A Sensing Field 4 bejárása során a pirossal jelölt robot később fejezi be az adatgyűjtést mint a társai, míg a Sensing Field 6 esetén a robotok közel egyforma idő alatt végeznek. A 4.17 ábrán a Sensing Field 4 mezőn a tervezett útvonal szintén jól elkülönülő, önmagát csak a kékkel jelölt útvonal metszi. A Sensing Field 6 mezőn az alútvonalak egymást többszörösen metszik. A kék útvonalban az utolsó Node-30 meglátogatása nagy plusz költséget jelent a robot számára, a sárga vagy a világos kékkel jelölt útvonalba való bevétele kisebb költséget jelenthetne. Ez a probléma a hangya kolónia algoritmus ciklusszám növelésével megoldható.

Az adatgyűjtési időt, azaz a leghosszabb alútvonal bejárási idejét a 6 érzékelő mezőn $k \in [1, 6]$ robotra a 4.19 ábra mutatja. Az adatgyűjtés ideje csökkenő tendenciát mutat az ágensek növekvő számával, azonban ez nem olyan szigorú, mint a periodikus adatgyűjtés esetén. Például a Sensing Field 4 és 5 esetén $k = 5$ robotot alkalmazva nagyobb a bejárási idő, mint $k = 4$ esetben, illetve a Sensing Field 2, 3 és 6 esetén $k = 6$ esetben nagyobb a bejárási idő, mint $k = 5$ robotot alkalmazva. Azonban a Sensing Field 1 esetén a bejárási idő szigorú monoton csökken az ágens szám növekedésével. Tehát a bejárási idő ebben az esetben nagyban függ a prioritásos csomópontok elhelyezkedésétől.

A robotok által összesen megtett útvonal hosszat a 4.20 ábra mutatja. Ez ebben az esetben is növekvő tendenciát mutat a növekvő ágens számmal. Az emelkedés azonban nem monoton a prioritásos csomópontok hamarabbi meglátogatása, és így önmagukat metsző útvonalak miatt.



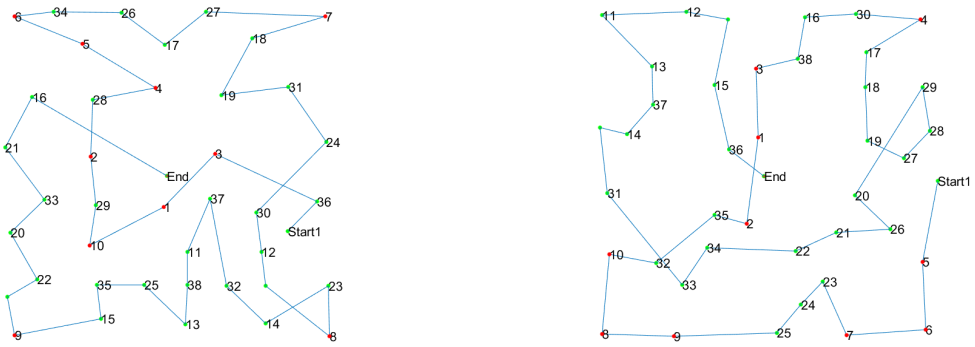
4.19. ábra. A hat különböző érzékelő mező bejárási ideje $k \in [1, 6]$ ágenst alkalmazva



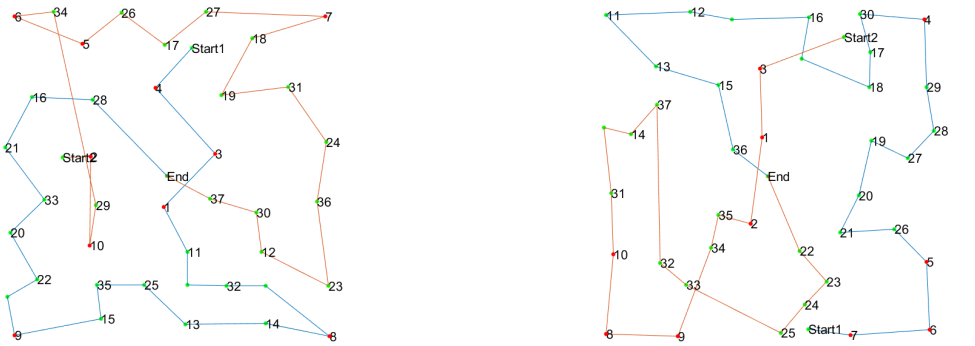
4.20. ábra. A hat különböző érzékelő mező összegzett úthossza $k \in [1, 6]$ ágenst alkalmazva

4.2.2. Véletlen helyről induló ágensek esete

Ebben a fejezetben annak az esetnek a szimulációs eredményei kerülnek bemutatásra, amikor az ágensek az érzékelő mező különböző, random módon kisorsolt pontjaiból indulnak, és az érzékelő mező közepén elhelyezkedő végpontba töltik fel az összegyűjtött adatokat. A szimulációt elvégeztem a hat különböző érzékelő mezőn $k \in [1, 6]$ ágenst alkalmazva. Az alútvonalak bejárási idejét a F.2.2 fejezet tartalmazza. A sensing Field 1 és Sensing Field 5 mezőkön való adatgyűjtés eredményét $k \in [1, 6]$ robotra a 4.21-4.26 ábrák mutatják, a prioritásos csomópontok pirossal jelöltek. A 4.21 ábrán mindkettő érzékelő

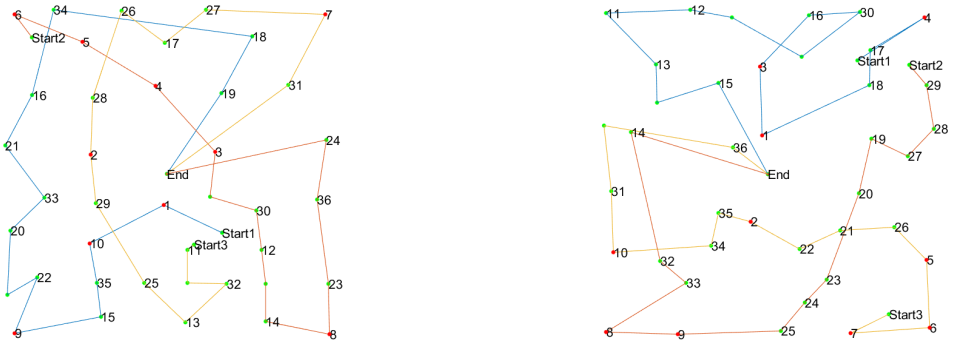


4.21. ábra. A Sensing Field 1 és Sensing Field 5 érzékelő mezők bejárása $k = 1$ ágenst alkalmazva.

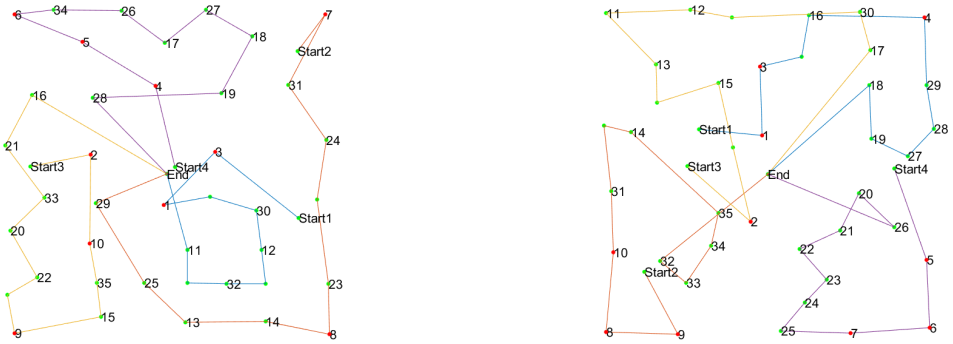


4.22. ábra. A Sensing Field 1 és Sensing Field 5 érzékelő mezők bejárása $k = 2$ ágenst alkalmazva.

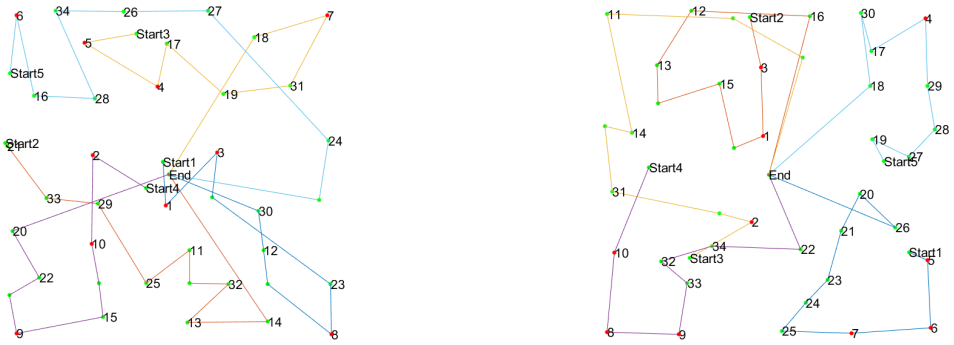
mezőre tervezett útvonal önmagát több helyen is metszi. A Sensing Field 1 esetén ez a prioritásos pontok hamarabbi bejárását segíti, míg a Sensing Field 5 esetében például a Node-27, Node-28, Node-29 és Node-19 csomópontok prioritás nélküli hurkot alkotnak, tehát az adatgyűjtés ideje nem optimális. A hangya kolónia algoritmus ciklus számának növelésével ez a hatás kiküszöbölhető. A 4.22 ábrán a Sensing Field 1 pirossal jelölt útvonalában a Node-35 nem prioritásos csomópont megelőzi a prioritásos Node-7 csomópontot, tehát ezek felcserélésével a bejárás javítható. Azonban a kékkel jelölt útvonal önmagát nem metszi, és a prioritásos csomópontok meglátogatását igyekszik előtérbe helyezni. A Sensing Field 5 pirossal jelölt útvonala önmagát metszi előtérbe helyezve ezzel a Node-9 és Node-10 prioritásos csomópontok hamarabbi meglátogatását. A kékkel jelölt útvonal 11. meglátogatott csomópontja után nem tartalmaz több prioritásos csomópontot. A 4.23 ábra tervezett útvonalai csak akkor tartalmaznak önmagukat metsző útvonalakat, ha az



4.23. ábra. A Sensing Field 1 és Sensing Field 5 érzékelő mezők bejárása $k = 3$ ágenst alkalmazva.

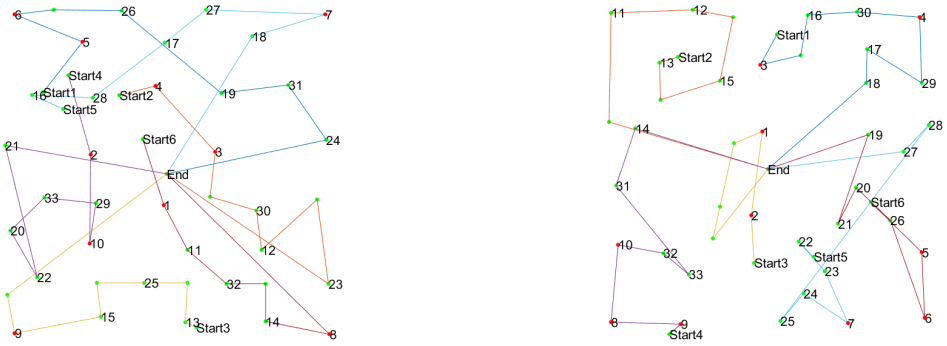


4.24. ábra. A Sensing Field 1 és Sensing Field 5 érzékelő mezők bejárása $k = 4$ ágenst alkalmazva.



4.25. ábra. A Sensing Field 1 és Sensing Field 5 érzékelő mezők bejárása $k = 5$ ágenst alkalmazva.

elősegíti a prioritásos csomópontok hamarabbi meglátogatását. A növekvő ágens számmal egyre több kezdőpont helyezkedik el a végpont környezetében, például 4.25 ábrán a Sensing Field 1 esetében. Ekkor önmagukat sokszorosan keresztező útvonalak keletkezhetnek. Az adatgyűjtést egyre több robottal végrehajtva látszólag egyre kaotikusabb útvonal keletkezik. Ennek az az oka, hogy ilyenkor a robotok igyekeznek előbb a prioritásos csomópontokhoz eljutni. Így növekvő ágens számmal a kezdő pontok véletlenszerű helyzete



4.26. ábra. A Sensing Field 1 és Sensing Field 5 érzékelő mezők bejárása $k = 6$ ágenst alkalmazva.

következtében egyre gyorsabban járják be a robotok a prioritással rendelkező csomópontokat. Több robotot alkalmazva előfordulhat, hogy egy robot nem jár be egyáltalán prioritásos csomópontot, hanem a prioritással nem rendelkező csomópontokból többet, amelyet a használt költség függvény (3.4) teljes mértékben támogat. Például 4.25 Sensing Field 1 második, pirossal jelölt útvonala. A 4.24 ábrán a Sensing Field 1 és Sensing Field 5 mezőre tervezett adatgyűjtési útvonalat láthatjuk $k = 4$ adatgyűjtő robot esetén. A bejárást a 5. és 6. videó (7 fejezet) mutatja. A Sensing Field 1 bejárása során a kék robot hamarabb beérkezik a bázispontba, míg a Sensing Field 5 sárgával jelölt ágense késlekedik a társaihoz képest.

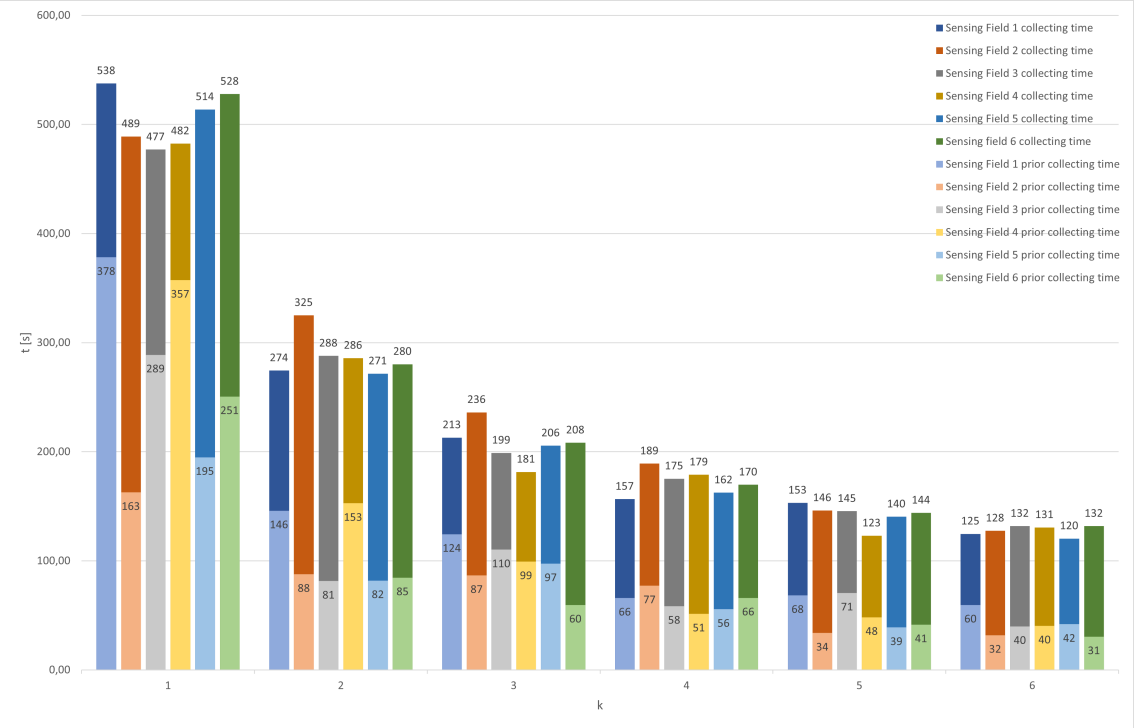
A 4.2 táblázat mutatja, hogy a teljes bejárási idő feléig, hány prioritással rendelkező csomópont került meglátogatásra a különböző érzékelő mezőkön $k \in [1, 6]$ ágenst alkalmazva. A Sensing Field 1 esetében a 10 prioritásos csomópontból 1, 2 és 3 ágenst alkalmazva 9 a többi esetben 10 csomópontot látogatnak meg a robotok a teljes bejárási idő feléig. A Sensing Field 5 esetén a teljes bejárási idő feléig az összes prioritásos csomópontot meglátogatják a robotok. Az is megfigyelhető, hogy $k = 5$ és $k = 6$ esetén már mindegyik érzékelő mezőn bejárásra kerül az összes prioritással rendelkező csomópont a bejárási idő feléig.

k	1	2	3	4	5	6
Sensing Field 1	9	9	9	10	10	10
Sensing Field 2	10	10	10	10	10	10
Sensing Field 3	10	10	9	10	10	10
Sensing Field 4	9	9	9	9	10	10
Sensing Field 5	10	10	10	10	10	10
Sensing Field 6	10	10	10	10	10	10

4.2. táblázat. A teljes bejárási idő feléig meglátogatott prioritásos csomópontok száma (10 prioritással rendelkező csomópontból)

A 4.27 ábrán látható az egyes érzékelő mezőkön az adatgyűjtés ideje, tehát az alútvonalak maximális bejárási ideje, és az összes prioritásos csomópont eléréséhez szükséges idő $k \in [1, 6]$ ágens esetén. Leolvasható, hogy a növekvő ágens számmal a prioritásos csomópontok bejárási ideje lényegesen lecsökken. Egy robotot alkalmazva az összes prioritással rendelkező csomópont meglátogatásának ideje közel két és fél illetve hat és fél perc között ingadozik, míg hat robotot alkalmazva minden esetben 1 perc alatti ez az érték. Az adatgyűjtés és a prioritásos csomópontok bejárásának az ideje a növekvő ágensszámmal egyre

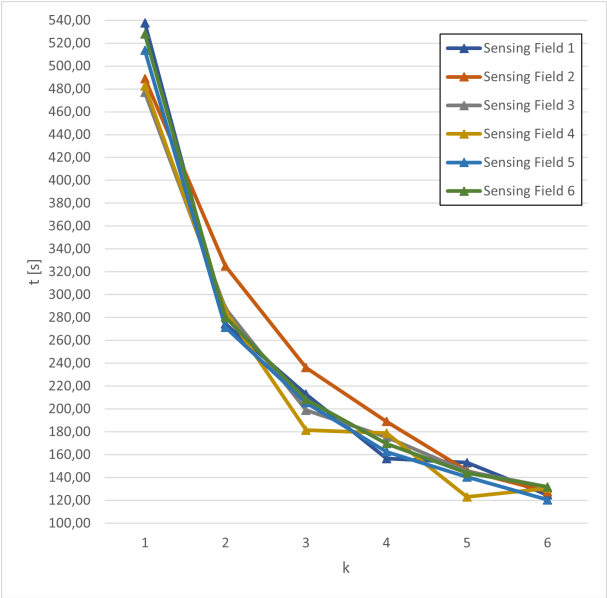
kevésbé csökken. Már négy robotot alkalmazva is bőven másfél perc alatt minden érzékelő mezőn meglátogatásra került az összes prioritással rendelkező csomópont.



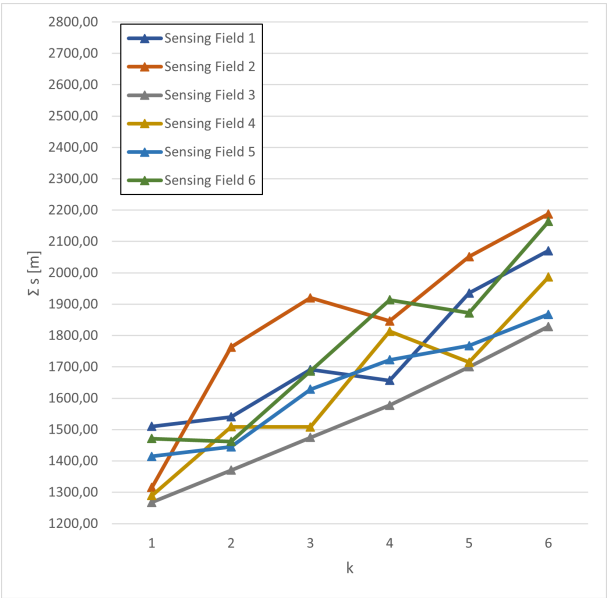
4.27. ábra. A hat különböző érzékelő mező teljes bejárási és az összes prioritásos csomópont meglátogatásához szükséges idő $k \in [1, 6]$ ágenst alkalmazva.

Az adatgyűjtési időt, azaz a leghosszabb alútvonal bejárási idejét a hat érzékelő mezőn $k \in [1, 6]$ robotra a 4.28 ábra mutatja. Növekvő ágens számmal a bejárási idő egyértelműen csökkenő tendenciát mutat. A csökkenés a Sensing Field 1 és Sensing Field 4 mezőkön nem monoton, valószínűleg a kezdőpontok szerencsétlen elhelyezkedése miatt. A Sensing Field 2 mező bejárási ideje kirívóan nagyobb mint a többi mezőé $k \in [2, 4]$ esetben a pálya elrendezésének következtében.

A robotok által összesen megtett útvonal hosszat a 4.29 ábra mutatja. Ez ebben az esetben is növekvő tendenciát mutat a növekvő ágens számmal. Azonban a kezdőpontok véletlen elhelyezkedéséből adódóan nem monoton.



4.28. ábra. A hat különböző érzékelő mező bejárási ideje prioritásos csomópontok között véletlen helyről induló $k \in [1, 6]$ ágenst alkalmazva

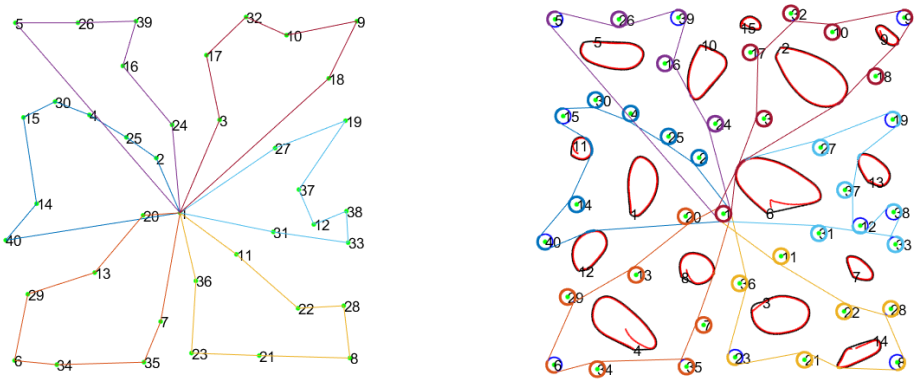


4.29. ábra. A hat különböző érzékelő mező összegzett úthossza prioritásos csomópontok között véletlen helyről induló $k \in [1, 6]$ ágenst alkalmazva

5. fejezet

Kitekintés

A legrövidebb út keresése egy gráfban, legyen az zárt, vagy nyitott kulcsfontosságú kérdés a robotok útvonalának megtervezésében. A [10] cikkben Dubins járművekkel történő adatgyűjtéshez készítettek algoritmust (SVPP–Shortest Viable Path Planning) akadályokat is tartalmazó érzékelő térben. Ennek első lépése az utazó ügynök, vagy a több eladós utazó ügynök probléma megoldása. Egy több szempontból is suboptimális megoldást ad, az akadályokat és az egykerekű Dubins robot tulajdonságait, csak az érzékelő csomópontok bejárási sorrendjének meghatározása után veszi figyelembe. A [29] dolgozatban több fajta algoritmust is bemutattam Dubins járművekből álló multiágensű rendszer útvonaltervezésére [10] algoritmusaiból kiindulva. A [10] eredményeit a [25] publikációban is felhasználtam a k-ACO algoritmus elkészítése során. Itt először a 2.2 fejezetben is kifejtett algoritmushoz hasonló, szintén általam készített MTSP-ACO algoritmust hajtom végre, majd pedig minden egyes részútvonalra az általam tovább fejlesztett SVPP [30] algoritmust futtatom le. Az 5.1 ábrán látható a MTSP megoldása hangya kolónia algoritmussal, illetve a felvázolt módon Dubins járműveknek készített útvonal, az alútvonal hosszakat a 5.1 táblázat tartalmazza.



5.1. ábra. A MTSP megoldása MTSP-ACO algoritmussal és a Dubins jármű számára tervezett adatgyűjtési út k-ACO algoritmussal akadályokat is tartalmazó érzékelési térben [25].

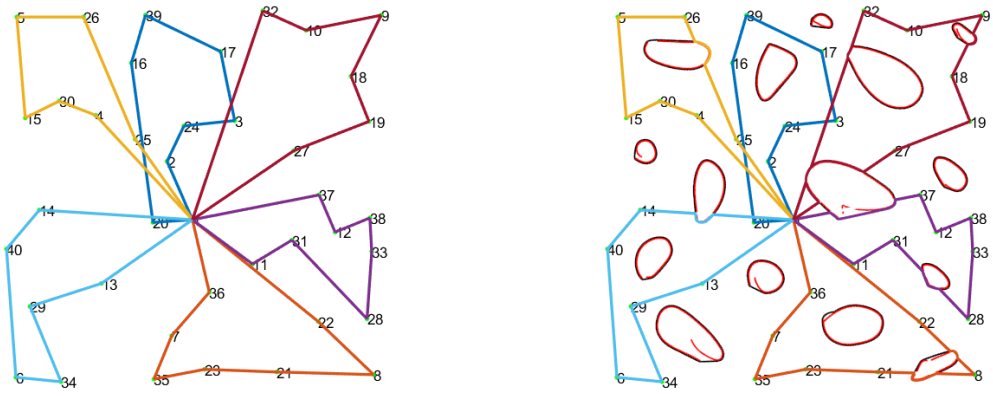
MTSP-ACO		k-ACO
282.1853		479.7463
288.0807		492.4063
297.2725		500.8628
314.9510		454.8688
286.3457		495.2448
325.0780		513.3965

5.1. táblázat. Az MTSP-ACO és a k-ACO algoritmusok által tervezett útvonalak hossza méterben.

A dolgozatomban bemutatott algoritmusok hasonlóképpen fejleszthetők tovább akadályokat is tartalmazó érzékelési térben való alkalmazásra. Az akadályok figyelembevétele történhet szuboptimális módon a csomópontok sorrendjének felállítása után, illetve előtt. Az első eset kisebb számítási kapacitást jelent, hiszen itt csak az egymásután sorrendben követő élek blokkoltságát kell vizsgálni majd az akadályok által blokkolt élek esetén az akadályok kerületén való végighaladás nagyságát figyelembe venni. A második esetben minden két csomópont között húzható él blokkoltságát figyelembe kell venni majd a megnövekedett úthosszt kiszámítani. Ez utóbbi esetben jobb megoldáshoz juthatunk, hiszen ekkor az algoritmus számít az akadályok által meghosszabbított élekre.

Az akadályok kikerülése többféleképpen történhet. A következőkben a legegyszerűbb triviális mód kerül bemutatásra. Ekkor ha egy él metsz egy akadályt, akkor a robot a kiinduló csomóponttól az első metszéspontig az élen, innen a második metszéspontig az akadály kerületén, majd a végcsomópontig szintén az élen halad.

Hat omnidirekcionális robotból álló multiágensű rendszer számára a tervezett útvonalat az 5.2 és 5.3 ábrák mutatják, az alutak hosszát a 5.2 táblázat foglalja össze. A 5.2 ábra első képén az akadályok nélküli útvonaltervezés eredményét láthatjuk míg a második az akadályok utólagos bevételét szemlélteti. Az akadályok több ízben is meghosszabbítják az előzetesen tervezett útvonalat, a bordóval jelölt leghosszabb alútvonal közel 25 méterrel nőtt meg.



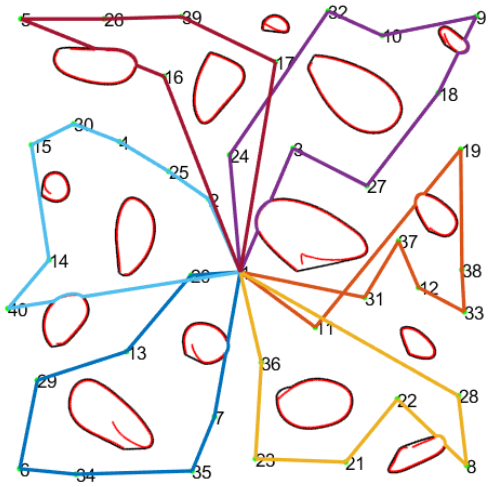
5.2. ábra. Útvonaltervezés Algoritmus 2 használatával akadályoktól eltekintve és az akadályok utólagos beszámítása.

Az 5.3 ábra az akadályok blokkoló hatásának előzetes figyelembe vételét szemlélteti. A 5.2 táblázatot tekintve az alúthosszak egymástól kisebb mértékben térnek el, mint az utólagosan bevett akadályok esetében, és a leghosszabb alútvonal közel 5 méterrel

szín	Akadályok nélkül	akadályok utólagos kezelése	akadályok előzetes kezelése
	282.1061	282.1061	288.2506
	322.7329	329.1630	317.1429
	312.8809	317.4584	316.0421
	272.0125	277.2891	355.3090
	326.6156	331.4106	283.0110
	335.9422	361.0374	342.9286

5.2. táblázat. A tervezett útvonal részútvonalainak hossza méterben az akadályok figyelembe vétele nélkül, akadályok utólagos figyelembe vételével, akadályok előzetes figyelembevételével.

rövidebb. Az 5.3 ábrán megfigyelhető, hogy a robotok a bejárás során az akadályokat elkerülik, vagy a lehető legkevesebbet tartózkodnak az akadályok kerületén.



5.3. ábra. Az Algoritmus 2 által készített útvonal az akadályok előzetes figyelembe vételével.

6. fejezet

Összegzés

A dolgozatomban omnidirekcionális robotokból álló multiágensű rendszer számára terveztem útvonalat érzékelő csomópontok között. A feladat az összes érzékelő csomóponton összegyűjtött adat feltöltése a bázis csomópontra. A minél hatékonyabb adatgyűjtés érdekében a cél a bejárandó útvonal és a bejárási idő csökkentése.

Először periodikus adatgyűjtéshez terveztem útvonalat, amikor cél a folyamatos adatgyűjtés egy érzékelő mezőről. Ekkor az ágensek egy előre kijelölt bázis csomópontból indulva az összes érzékelő csomópontról együttesen begyűjtik az adatot, majd a bázis csomópontra visszatérve feltöltik.

A megoldást először elkészítettem egy adatgyűjtő robot számára, új hangya kolónia optimalizáláson alapuló algoritmust készítettem az utazó ügynök probléma megoldására. Ezt követően az algoritmust kiterjesztettem több ágensű utazó probléma megoldására, amelynek során új költség függvényt definiáltam.

A harmadik fejezetben prioritással is rendelkező csomópontok közötti egyszeri adatgyűjtéshez készítettem hangya kolónia optimalizáción alapuló új algoritmust. Ennek során új költség függvényt definiáltam, és a feromon frissítésre is új módszert fejlesztettem ki.

A dolgozatomban részletesen tárgyaltam az általam készített algoritmusok működését. Az útvonal tervezés eredményeit bemutattam mind a periodikus, mind az egyszeri prioritással rendelkező csomópontokról való adatgyűjtés esetére is. Hat különböző szimulációs mezőn különböző számú ágenseket alkalmazva végeztem szimulációkat. Az eredményeket mind numerikusan mind vizuálisan ismertettem. Több különböző szempontból, különböző mérőszámokat használva összehasonlítottam. Vizsgáltam a pálya egészének bejárásához szükséges időt, a robotok által megtett úthosszak összegét, illetve az átlagos eltérését az egyes robotok által megtett útvonalaknak, amely az adatok késleltetési idejének eltérésére mutatott rá. Illetve felvázoltam, hogy az egyes algoritmusokban a mérőszámaik hogyan változnak az alkalmazott robotok számának függvényében.

Az egyszeri prioritásos csomópontokat is tartalmazó adatgyűjtés elemzéséhez új mérőszámokat definiáltam. Vizsgáltam, hogy mennyi idő szükséges az összes prioritásos csomópont bejárásához, illetve, hogy a bejárási idő 50%-ig mennyi prioritásos csomópont kerül meglátogatásra. A könnyebb átláthatóság, elemzés érdekében a numerikus eredményeket grafikonokon szemléltettem.

A dolgozatomat kitekintéssel zártam, ahol bemutattam az általam fejlesztett algoritmusok alkalmazását akadályokat is tartalmazó érzékelési térben való útvonaltervezésre.

A periodikus útvonaltervezés, az önmagukat metsző útvonalak miatt még nem teljes, ennek megoldása hangya algoritmussal történő optimalizálással történhet. A prioritásos csomópontok közötti adatgyűjtés megoldása során célszerű új költség függvényt definiálni, amely bonyolultabb prioritási rendszerrel rendelkező területen is képes optimális útvona-

lat tervezni. A megoldás általánosítható akadályokat is tartalmazó érzékelési térben való adatgyűjtéshez, ez a későbbi munkám célkitűzése.

Összefoglalva, az általam készített algoritmusok jó megoldást adnak mind periodikus, mind pedig prioritásos csomópontokat tartalmazó egyszeri adatgyűjtéshez.

7. fejezet

Videók

A videók a https://bmeedu-my.sharepoint.com/:f:/g/personal/olasz-szabo_sara_edu_bme_hu/EmTsA5rsCq1MixXUpx4i634BNqGyaDUgl51-4lgQekZcZw?e=2thcHL címen érhetők el.

Köszönetnyilvánítás

A Kulturális és Innovációs Minisztérium **ÚNKP-22-2-III-BME-233** kódszámú Új Nemzeti Kiválóság Programjának a Nemzeti Kutatási, Fejlesztési és Innovációs Alapból finanszírozott szakmai támogatásával készült.

Szeretném megköszönni a konzulensemnek, Dr. Harmati István Tanár Úrnak a rengeteg segítséget, a gyors és pontos válaszokat a kérdéseimre, a fantasztikus ötleteit és a rendszeres konzultációkat, amely nélkül ez a munka nem készülhetett volna el.

Irodalomjegyzék

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless sen- sor networks for habitat monitoring, in: The 1st ACM International Workshop on Wireless Sensor Networks and Applications, 2002, pp. 88–97

[2] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, B. Krogh, Energy-efficient surveillance system using wireless sen- sor networks, in: the 2nd International Conference on Mobile Systems, Appli- cations, and Services, ACM, 2004, pp. 270–283

[3] J. N. Al-Karaki, A. E. Kamal, Routing techniques in wireless sensor networks: a survey, IEEE Wireless Communications, vol.11, no. 6, 2004, pp. 6-28. DOI: 10.1016/j.proeng.2012.06.320

[4] Gu, Y., Ren, F., Ji, Y., Li, J. (2016). The Evolution of Sink Mobility Management in Wireless Sensor Networks: A Survey. IEEE Communications Surveys and Tutorials, 18(1), 507–524. doi:10.1109/comst.2015.2388779

[5] I. Chatzigiannakis, A. Kinalis, S. Nikoletseas, Sink mobility protocols for data collection in wireless sensor networks, in: the 4th ACM International Work- shop on Mobility Management and Wireless Access, ACM, 2006, pp. 52–59

[6] Y. Yun, Y. Xia, Maximizing the lifetime of wireless sensor networks with mo- bile sink in delay-tolerant applications, IEEE Trans. Mob. Comput. 9 (9) (2010) 1308–1318

[7] Hailong Huang, Andrey V. Savkin, Ming Ding, Chao Huang, Mobile robots in wireless sensor networks: A survey on tasks, Computer Networks 148, 2019, pp. 1-19. DOI: 10.1016/j.comnet.2018.10.018

[8] Huang, Hailong, Andrey V. Savkin, Reactive 3D deployment of a flying robotic network for surveillance of mobile targets, Computer Networks 161, 2019, pp.172-182. DOI: 10.1016/j.comnet.2019.06.020

[9] H. Huang, A. V. Savkin, An energy efficient approach for data collection in wireless sensor networks using public transportation vehicles, AEU-International Journal of Electronics and Communications 75, 2017, pp. 108-118. DOI: 10.1016/j.aeue.2017.03.012

[10] Hailong Huang, Andrey V. Savkin, Viable path planning for data collection robots in a sensing field with obstacles, Computer Communications 111 (2017) 84–96

[11] Taheri, Hamid, and Chun Xia Zhao. "Omnidirectional mobile robots, mechanisms and navigation approaches." Mechanism and Machine Theory 153 (2020): 103958.

[12] X. Ren, W. Liang, W. Xu, Data collection maximization in renewable sensor networks via time-slot scheduling, IEEE Trans. Comput. 64 (7) (2015) 1870–1883

[13] Cheikhrouhou, Omar, and Ines Khoufi. "A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy." Computer Science Review 40 (2021): 100369.

[14] JUNJIE, Pan; DINGWEI, Wang. An ant colony optimization algorithm for multiple travelling salesman problem. In: First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06). IEEE, 2006. p. 210-213.rategy. 1991.

[15] YANG, Jinhui, et al. An ant colony optimization method for generalized TSP problem. Progress in Natural Science, 2008, 18.11: 1417-1422.

[16] HINGRAJIYA, Krishna H.; GUPTA, Ravindra Kumar; CHANDEL, Gajendra Singh. An Approach for Solving Multiple Travelling Salesman Problem using Ant Colony Optimization. In: Fourth International conference on Computer engineering and Intelligence system. 2015. p. 1133-1149.

[17] RAMADHANI, T.; HERTONO, Gatot Fatwanto; HANDARI, Bevina Desjwiandra. An Ant Colony Optimization algorithm for solving the fixed destination multi-depot multiple traveling salesman problem with non-random parameters. In: AIP Conference Proceedings. AIP Publishing LLC, 2017. p. 030123.

[18] LU, Li-Chih; YUE, Tai-Wen. Mission-oriented ant-team ACO for min–max MTSP. Applied Soft Computing, 2019, 76: 436-444.

[19] Doan, T. T., Bostel, N., Hà, M. H. (2021). The vehicle routing problem with relaxed priority rules. EURO Journal on Transportation and Logistics, 10, 100039.

[20] M.H., Nguyen Phuong, H., Tran Ngoc Nhat, H., Langevin, A., Trepanier, M., 2020. Solving the clustered traveling salesman problem with -relaxed priority rule. Int. Trans. Oper. Res. (in press).

[21] Panchamgam, K.V., 2011. Essays in Retail Operations and Humanitarian Logistics. Ph.D. thesis. Robert H.Smith School of Business, University of Maryland, College Park, Md.

[22] Shetty, V.K., Sudit, M., Nagi, R., 6, 2008. Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles. Comput. Oper. Res. 35 (6), 1813–1828, 6.

[23] DORIGO, Marco; MANIEZZO, Vittorio; COLORNI, Alberto. Positive feedback as a search st

[24] DORIGO, Marco; GAMBARDELLA, Luca Maria. Ant colonies for the travelling salesman problem. biosystems, 1997, 43.2: 73-81.

[25] Sára Olasz-Szabó, István Harmati. Path planning for data collection multiagent system in a sensing field with obstacles. 2022 25th International Symposium on Measurement and Control in Robotics

[26] G. A. CROES A method for solving traveling salesman problems. Operations Res. 6 (1958) , pp., 791-812.

[27] Stützle, Thomas, and Marco Dorigo. ACO algorithms for the traveling salesman problem. Evolutionary algorithms in engineering and computer science 4 (1999): 163-183.

[28] GAERTNER, Dorian; CLARK, Keith L. On Optimal Parameters for Ant Colony Optimization Algorithms. In: IC-AI. 2005. p. 83-89.

[29] Olasz-Szabó Sára. Adatgyűjtő robotok pályatervezése érzékelő csomópontok között. BME-VIK Tudományos Diákköri Konferencia 2021. https://tdk.bme.hu/VIK/inteli/Adatgyujto-robotok-palyatervezese-erzekelo (2022.10.26)

[30] Sára Olasz-Szabó, István Harmati. Path planning for data collection robot in sensing field with obstacles. Acta IMEKO Vol. 11 No. 3 (2022) DOI: https://doi.org/10.21014/acta_imeko.v11i3.1254

Függelék

F.1. Periodikus adatgyűjtés eredménye

k								
1	474,93							
2	242,26	249,87						
3	181,70	169,50	187,54					
4	152,10	150,02	151,03	144,19				
5	105,51	128,12	126,17	119,55	118,09			
6	104,74	93,65	100,68	119,77	115,87	115,36		
7	85,25	92,05	97,74	98,66	102,31	105,34	105,27	
8	71,56	95,94	93,31	86,81	102,42	98,36	91,92	97,33

F.1. táblázat. Sensing Field 1 rész útvonalak bejárási ideje [s] periodikus bejárás során

k								
1	449,24							
2	248,48	253,71						
3	181,61	180,87	172,22					
4	150,71	143,56	157,06	152,36				
5	129,18	134,08	117,73	127,75	137,23			
6	120,01	116,74	95,71	117,51	122,00	119,84		
7	104,09	77,59	107,45	109,85	118,70	117,79	115,36	
8	99,44	111,81	99,11	94,57	66,13	109,41	102,48	111,76

F.2. táblázat. Sensing Field 2 rész útvonalak bejárási ideje [s] periodikus bejárás során

k								
1	471,49							
2	251,38	251,93						
3	184,29	166,75	175,40					
4	146,24	138,42	142,17	150,47				
5	101,88	119,06	123,09	123,48	122,44			
6	105,20	95,08	103,94	111,81	110,22	108,25		
7	99,16	103,55	90,55	105,58	96,23	104,81	88,33	
8	95,39	96,03	95,60	86,05	92,87	103,05	99,24	100,37

F.3. táblázat. Sensing Field 3 rész útvonalak bejárási ideje [s] periodikus bejárás során

k								
1	460,60							
2	248,73	255,20						
3	176,88	170,56	175,97					
4	129,49	143,94	137,89	137,71				
5	115,87	120,87	104,29	130,31	128,45			
6	107,34	104,42	107,89	111,45	109,04	101,56		
7	105,71	98,13	96,15	94,36	107,02	108,60	102,22	
8	81,63	93,34	93,44	85,28	102,47	92,47	98,84	83,59

F.4. táblázat. Sensing Field 4 rész útvonalak bejárási ideje [s] periodikus bejárás során

k								
1	464,77							
2	255,30	251,39						
3	188,07	189,27	185,78					
4	151,18	151,19	140,97	134,76				
5	124,38	124,96	115,69	129,16	126,92			
6	97,17	103,86	114,25	100,25	116,01	115,03		
7	107,66	90,33	107,28	101,49	84,04	101,44	105,72	
8	79,20	87,00	77,52	95,53	93,95	97,59	99,89	99,12

F.5. táblázat. Sensing Field 5 rész útvonalak bejárási ideje [s] periodikus bejárás során

k								
1	451,61							
2	241,12	242,88						
3	186,71	147,75	187,70					
4	148,40	132,60	141,99	148,28				
5	120,73	129,02	119,58	121,60	121,48			
6	108,39	113,94	107,88	110,31	115,21	112,54		
7	106,79	106,69	101,41	94,05	101,30	98,45	104,01	
8	99,33	101,74	95,32	91,44	89,95	94,56	99,16	99,18

F.6. táblázat. Sensing Field 6 rész útvonalak bejárási ideje [s] periodikus bejárás során

Sensing Field 1			
k	Teljes bejárási idő [s]	Átlagtól való átlagos eltérés [s]	Összegzett úthossz [m]
1	474,93	0,00	1275,74
2	249,87	3,80	1344,53
3	187,54	6,72	1530,97
4	152,10	2,57	1765,35
5	128,12	6,15	1765,82
6	119,77	8,65	1976,30
7	105,34	5,49	2122,49

F.7. táblázat. Sensing Field 1, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz periodikus bejárás során

Sensing Field 2			
k	Teljes bejárási idő [s]	Átlagtól való átlagos eltérés [s]	Összegzett úthossz [m]
1	449,24	0,00	1172,96
2	253,71	2,61	1384,76
3	181,61	4,01	1514,78
4	157,06	3,79	1790,77
5	137,23	5,17	1959,91
6	122,00	6,53	2143,29
7	118,70	9,38	2379,29
8	111,81	9,55	2554,86

F.8. táblázat. Sensing Field 2, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz periodikus bejárás során

Sensing Field 3			
k	Teljes bejárási idő [s]	Átlagtól való átlagos eltérés [s]	Összegzett úthossz [m]
1	471,49	0,00	1261,97
2	251,93	0,27	1389,27
3	184,29	5,87	1481,78
4	150,47	4,03	1685,20
5	123,48	6,45	1735,78
6	111,81	4,34	1914,03
7	105,58	5,67	2128,89
8	103,05	3,61	2450,37

F.9. táblázat. Sensing Field 3, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz periodikus bejárás során

Sensing Field 4			
k	Teljes bejárási idő [s]	Átlagtól való átlagos eltérés [s]	Összegzett úthossz [m]
1	460,60	0,00	1218,41
2	255,20	3,24	1391,72
3	176,88	2,61	1469,63
4	143,94	3,88	1572,16
5	130,31	7,90	1775,15
6	111,45	2,64	1942,77
7	108,60	4,74	2224,73
8	102,47	5,91	2300,21

F.10. táblázat. Sensing Field 4, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz periodikus bejárás során

Sensing Field 5			
k	Teljes bejárási idő [s]	Átlagtól való átlagos eltérés [s]	Összegzett úthossz [m]
1	464,77	0,00	1235,08
2	255,30	1,96	1402,76
3	189,27	1,29	1628,44
4	151,19	6,66	1688,40
5	129,16	3,41	1860,42
6	116,01	7,33	1962,34
7	107,66	7,16	2167,87
8	99,89	7,49	2295,27

F.11. táblázat. Sensing Field 5, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz periodikus bejárás során

Sensing Field 6			
k	Teljes bejárási idő [s]	Átlagtól való átlagos eltérés [s]	Összegzett úthossz [m]
1	451,61	0,00	1182,43
2	242,88	0,88	1311,98
3	187,70	17,54	1464,63
4	148,40	5,52	1661,09
5	129,02	2,61	1825,67
6	115,21	2,52	2049,05
7	106,79	3,44	2226,81
8	101,74	3,52	2458,76

F.12. táblázat. Sensing Field 6, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz periodikus bejárás során

F.2. Prioritásos csomópontokról való adatgyűjtés esete

F.2.1. Egy kezdőpontból induló ágensek esete

k						
1	524,72					
2	268,05	288,10				
3	204,59	182,03	185,22			
4	161,59	147,04	176,63	129,92		
5	126,23	109,80	148,39	146,40	149,81	
6	132,90	105,52	111,63	134,71	105,40	130,46

F.13. táblázat. Sensing Field 1 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn.

k						
1	500,77					
2	297,89	310,02				
3	209,17	196,08	267,77			
4	162,33	104,59	207,62	148,72		
5	159,19	124,17	137,76	148,17	128,93	
6	110,81	140,43	143,59	101,86	181,19	141,76

F.14. táblázat. Sensing Field 2 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn.

k						
1	534,78					
2	312,22	320,55				
3	209,20	184,51	144,84			
4	187,44	131,27	153,38	164,01		
5	129,67	133,54	128,83	119,30	149,04	
6	115,33	141,36	125,23	118,39	113,77	150,98

F.15. táblázat. Sensing Field 3 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn.

k						
1	518,83					
2	274,92	265,08				
3	205,80	214,19	170,65			
4	146,85	169,16	150,99	141,88		
5	146,35	165,94	155,41	127,48	102,12	
6	123,17	128,92	136,73	138,45	105,40	120,37

F.16. táblázat. Sensing Field 4 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn.

k						
1	524,64					
2	289,93	276,81				
3	222,70	187,25	198,33			
4	160,54	149,05	131,43	159,21		
5	130,12	163,36	146,24	127,85	122,69	
6	115,89	115,36	139,09	115,77	152,20	119,15

F.17. táblázat. Sensing Field 5 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn.

k						
1	487,80					
2	240,31	289,67				
3	187,67	158,04	226,44			
4	150,21	145,83	121,71	148,38		
5	124,24	136,88	127,23	131,67	127,35	
6	150,82	134,52	107,84	114,17	115,21	120,63

F.18. táblázat. Sensing Field 6 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn.

Sensing Field 1			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	524,72	332,72	1458,87
2	288,10	223,60	1584,62
3	204,59	154,75	1647,33
4	176,63	105,46	1820,74
5	149,81	93,33	2082,54
6	134,71	93,17	2242,48

F.19. táblázat. Sensing Field 1, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során

Sensing Field 2			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	500,77	182,48	1363,06
2	310,02	138,61	1791,66
3	267,77	160,85	2052,10
4	207,62	122,51	1852,99
5	159,19	119,25	2152,86
6	181,19	93,65	2638,52

F.20. táblázat. Sensing Field 2, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során

Sensing Field 3			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	534,78	160,54	1499,11
2	320,55	123,19	1891,09
3	209,20	145,60	1514,19
4	187,44	130,17	1904,36
5	149,04	127,83	2001,54
6	150,98	118,49	2420,25

F.21. táblázat. Sensing Field 3, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során

Sensing Field 4			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	518,83	386,58	1435,31
2	274,92	194,42	1520,00
3	214,19	139,86	1722,51
4	169,16	121,85	1795,50
5	165,94	85,11	2149,22
6	138,45	103,37	2372,14

F.22. táblázat. Sensing Field 1, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során

Sensing Field 5			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	524,64	246,47	1458,55
2	289,93	174,02	1626,97
3	222,70	153,68	1793,13
4	160,54	123,85	1760,91
5	163,36	101,12	2121,03
6	152,20	95,79	2389,83

F.23. táblázat. Sensing Field 5, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során

Sensing Field 6			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	487,80	248,09	1311,20
2	289,67	86,84	1479,89
3	226,44	81,79	1648,62
4	150,21	64,59	1624,52
5	136,88	60,13	1949,45
6	150,82	53,74	2332,70

F.24. táblázat. Sensing Field 6, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során

F.2.2. Véletlen helyről induló ágensek esete

k						
1	537,53					
2	270,77	274,36				
3	212,95	182,40	187,52			
4	113,74	149,59	154,35	156,55		
5	120,20	130,31	120,65	119,58	153,07	
6	124,71	116,63	100,61	123,78	109,87	102,05

F.25. táblázat. Sensing Field 1 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn random Start pontok esetén.

k						
1	488,92					
2	275,72	325,04				
3	209,46	194,51	236,10			
4	189,06	143,98	126,07	162,50		
5	146,10	120,60	139,60	137,58	129,03	
6	127,51	113,51	110,05	116,53	121,20	118,19

F.26. táblázat. Sensing Field 2 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn random Start pontok esetén.

k						
1	476,92					
2	214,76	287,84				
3	194,14	198,89	135,70			
4	175,27	100,38	172,10	106,73		
5	145,43	107,23	118,39	131,66	82,38	
6	83,17	71,09	131,79	128,06	82,25	120,82

F.27. táblázat. Sensing Field 3 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn random Start pontok esetén.

k						
1	482,37					
2	285,81	251,44				
3	178,68	181,46	176,88			
4	178,89	147,48	146,60	140,31		
5	108,54	119,58	121,24	116,41	122,97	
6	115,50	97,83	111,34	130,58	100,74	100,74

F.28. táblázat. Sensing Field 4 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn random Start pontok esetén.

k						
1	513,71					
2	271,50	249,75				
3	205,51	187,05	174,64			
4	141,75	146,90	162,46	139,50		
5	119,57	127,56	140,41	107,10	107,32	
6	109,30	113,82	71,49	109,93	120,37	101,94

F.29. táblázat. Sensing Field 5 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn random Start pontok esetén.

k						
1	527,82					
2	280,21	245,27				
3	196,93	208,19	176,54			
4	169,69	166,14	139,60	163,00		
5	123,80	115,71	110,33	134,27	143,93	
6	107,71	91,13	121,70	130,23	118,41	131,77

F.30. táblázat. Sensing Field 6 rész útvonalak bejárási ideje [s] prioritásos csomópontokat tartalmazó mezőn random Start pontok esetén.

Sensing Field 1			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	537,53	378,20	1510,10
2	274,36	145,91	1540,53
3	212,95	124,36	1691,46
4	156,55	65,93	1656,95
5	153,07	68,34	1935,23
6	124,71	59,56	2070,60

F.31. táblázat. Sensing Field 1, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során random Start pontokból induló ágensek esetén

Sensing Field 2			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	488,92	162,76	1315,70
2	325,04	87,66	1763,04
3	236,10	86,62	1920,27
4	189,06	77,19	1846,43
5	146,10	33,79	2051,65
6	127,51	31,86	2187,97

F.32. táblázat. Sensing Field 2, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során random Start pontokból induló ágensek esetén

Sensing Field 3			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	476,92	288,82	1267,68
2	287,84	81,43	1370,41
3	198,89	110,19	1474,95
4	175,27	58,45	1577,90
5	145,43	70,59	1700,36
6	131,79	39,92	1828,68

F.33. táblázat. Sensing Field 3, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során random Start pontokból induló ágensek esetén

Sensing Field 4			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	482,37	357,34	1289,49
2	285,81	152,71	1508,98
3	181,46	99,26	1508,05
4	178,89	51,40	1813,12
5	122,97	48,23	1714,92
6	130,58	40,40	1986,91

F.34. táblázat. Sensing Field 4, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során random Start pontokból induló ágensek esetén

Sensing Field 5			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	513,71	194,90	1414,83
2	271,50	81,91	1445,00
3	205,51	97,45	1628,78
4	162,46	55,58	1722,41
5	140,41	38,91	1767,85
6	120,37	41,84	1867,42

F.35. táblázat. Sensing Field 5, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során random Start pontokból induló ágensek esetén

Sensing Field 6			
k	Teljes bejárási idő [s]	Prioritásos csomópontok bejárási ideje [s]	Összegzett úthossz [m]
1	527,82	250,61	1471,30
2	280,21	84,54	1461,92
3	208,19	59,54	1686,62
4	169,69	66,04	1913,74
5	143,93	41,48	1872,16
6	131,77	30,51	2163,84

F.36. táblázat. Sensing Field 6, teljes bejárási idő, a bejárási idők átlagtól vett átlagos eltérése és az összegzett úthossz prioritásos csomópontokat is tartalmazó egyszeri bejárás során random Start pontokból induló ágensek esetén