Collision-Free Robot Scheduling

Nick Mónika, Jurca Henriette, Szilágyi Csaba

Eredeti szerzők

Duncan Adamson, Nathan Flaherty, Igor Potapov, Paul G. Spirakis

[Abstract 2](#_Toc183966033)

[Bevezetés 2](#_Toc183966034)

[Eredmények 4](#_Toc183966035)

[K-approximációs algoritmus 4](#_Toc183966036)

[Dinamikus programozási algoritmus 4](#_Toc183966037)

[Az elkészített program 5](#_Toc183966038)

[Egyedi szituáció létrehozása 5](#_Toc183966039)

[Generált gráf 6](#_Toc183966040)

[Program futása 7](#_Toc183966041)

[Szimulációk 8](#_Toc183966042)

[Konklúzió 11](#_Toc183966043)

# Abstract

Napjainkban a robotok szerves részét képezik a gyártósoroknak, kutató laboratóriumoknak. Azonban ahogy fejlődik a technológia, úgy jelennek meg a mindennapi élet egyre több területén is. Annak érdekében, hogy ezek a robotok időérzékeny környezetben is a maximumot tudják nyújtani, meg kell vizsgálni, hogy hogyan lehet a számukra kiosztott feladatot a legoptimálisabban ütemezni.

A kutatásban laboratóriumban elhelyezkedő, fix helyen elvégzendő feladatok lebonyolítására kitalált robotokon fogunk kísérletezni. A laboratóriumot egy gráf jelképezi, ahol a feladatok a csúcsok. A csúcsokat összekötő élek jelzik az utat, amit követni kell annak érdekében, hogy egyik feladatból a másikba jussunk. Minden feladatnak van egy elvégzési ideje, melyet amennyiben egy robot megkezd, úgy köteles végig csinálni. Az élek is rendelkeznek egy időtartammal, amely azt mutatja meg, hogy hány időegység, mire egyik csúcsból a másikba jut a robot. Továbbá megkötés, hogy egy csúcson nem helyezkedhet el két robot egy időben.

A cél, hogy minden a gráfon elhelyezkedő robot számára megállapítsunk egy olyan ütemtervet, mely optimalizálja az ütemtervét annak a robotnak, amely a legtöbb időegység alatt fejezi be az összes, számára kijelölt feladatot.

A kutatásban megmutatjuk, hogy ez a probléma több szimpla gráf esetén is NP-teljes. Egészen pontosan ilyen gráfok a teljes gráfok, bipartit, csillag és síkgráfok. Pozitív eredményeket mutatunk azonban útgráfok esetén, ahol optimális ütemterv készletet tudunk megállapítani robotnak, feladattal és egyenlő hosszúságú feladatok esetén, idő alatt. Továbbá egy k-approximációs algoritmust készítünk arra az esetre, ha a feladat hossza nem egyenlő mindenhol.

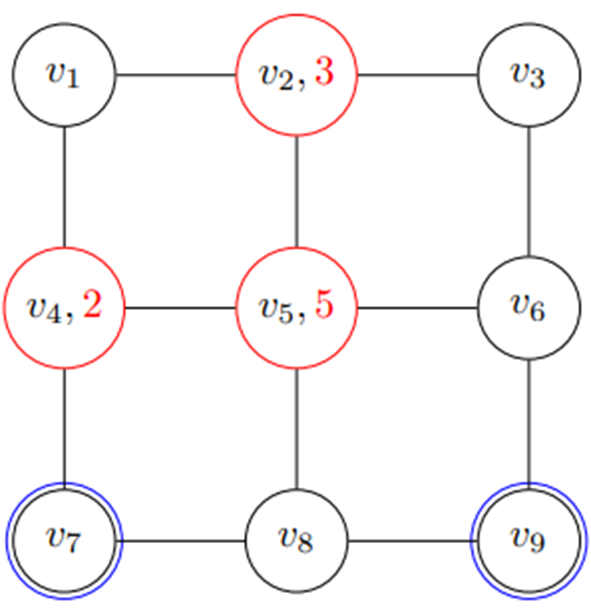
# Bevezetés

A robotok ütemezése és mozgásának optimalizálása egyre fontosabb szerepet játszik a modern automatizálási rendszerekben. A tanulmányban példaképpen egy laboratóriumi környezetet vettünk alapul, ahol a robotok meghatározott feladatokat végeznek el előre definiált helyszíneken. E tanulmányban a többrobotos ütemezési problémát vizsgálták a szerzők, amely a feladatok hatékony elvégzésére és az ütközések elkerülésére koncentrál. A probléma gráfelméleti modellezésén keresztül olyan algoritmusokat fejlesztettek ki az eredeti szerzők, amelyek a lehető leggyorsabb feladatvégzést biztosítják, miközben a robotok mozgását összehangolt módon szabályozzák.

A gráfokban a robotok mozgását az alábbi feltételek szabályozzák:

* Minden robotot a gráf egy adott csúcsára (pl. ) helyezünk.
* Minden feladat egy adott csúcsban található (), és az elvégzéséhez egy robotnak ott kell tartózkodnia meghatározott időtartamig ().
* A cél egy olyan ütemterv () megalkotása, amely minimalizálja az összes robot által szükséges maximális időtartamot (), ahol az -edik robot ütemterve).

Annak érdekében, hogy a problémát jobban ellehessen képzelni, az alábbi ábra jól szemlélteti egy 9 csúcsból álló gráfon a laboratóriumban található feladatokat.

A pirossal jelölt csomópontok az elvégzendő feladatokat ( ) jelölik, például , amely a csúcson található feladatot és annak 3 egységnyi időtartamát mutatja. A kék körvonalú csúcsok jelölik a robotok indulási helyzetét ( és ). Ez az ábra a robotok és a feladatok kezdeti elhelyezkedését ábrázolja egy laboratóriumi vagy általános munkakörnyezet modellezésére.

A probléma bonyolultsága függ az alkalmazott gráftól. Ennek oka, hogy a gráf szerkezete meghatározza a robotok mozgásának és az ütközések elkerülésének lehetőségeit. Például teljes gráfokban a robotok bármely csúcsra könnyen eljuthatnak, ami növeli a kombinatorikai lehetőségeket, és így a komplexitást, míg útgráfokban a mozgás korlátozott, ami egyszerűbb algoritmusokat tesz lehetővé.

* Általános gráfok esetén, ahol robot van, a probléma NP-teljes.
* Az útgráfokra optimalizált algoritmus érhető el időkomplexitással, ahol a robotok száma, a feladatok száma, és a gráf mérete.
* Ha a feladatok hosszúsága nem kötött, k-approximációs algoritmus alkalmazható.

|  |  |
| --- | --- |
| Beállítás | Eredmény |
| Általános gráfok, | NP-teljes |
| Teljes gráfok, | NP-teljes |
| Bipartit gráfok, | NP-teljes |
| Csillag gráfok, | NP-teljes |
| Síkgráfok, | NP-teljes |
| Útgráfok egyenlő feladat hosszal, | Optimális idő O(kmn) algoritmus |
| Útgráfok, | k-approximációs algoritmus |

A tanulmány célja, hogy megértse a problémakör nehézségeit különböző gráfosztályokon, miközben gyakorlati megoldásokat nyújt, különösen az egyszerűbb, lineáris gráfok esetén.

# Eredmények

A kutatás során különböző gráftípusokra vonatkozóan értékeltük a robotok ütemezésének problémáját. Megállapítottuk, hogy a probléma általános esetben NP-teljes, azonban bizonyos egyszerűsített gráfosztályok, például útgráfok esetén hatékony megoldások találhatók. Ezen eredmények alapján optimalizált és approximációs algoritmusokat teszteltünk, amelyek jelentősen csökkentik az ütemezési időt és a számítási komplexitást.

## K-approximációs algoritmus

Olyan approximációs algoritmust dolgoztak ki a szerzők, amely garantáltan legfeljebb -szor lassabb megoldást biztosít az optimális megoldáshoz képest, ahol a robotok száma. Ez a megközelítés különösen akkor hasznos, ha a feladatok hosszúsága változó, és az optimális megoldás megtalálása számítási szempontból túl költséges lenne. Az algoritmus egyszerűségének köszönhetően nagyobb méretű gráfokon és több robot esetén is alkalmazható.

## Dinamikus programozási algoritmus

Az útgráfokra egy dinamikus programozási megoldást teszteltünk, amely optimális ütemezést biztosít egyenlő hosszúságú feladatok esetén. Az algoritmus a gráf feladatait részekre bontja, és rekurzívan meghatározza a legrövidebb időtartamot biztosító megoldást. Ez a megközelítés időkomplexitással dolgozik, ahol a robotok száma, a feladatok száma, és a gráf csúcspontjainak száma. Az eredmény garantáltan optimális, így ideális kisebb és közepes méretű problémák megoldására.

# Az elkészített program

A készített demonstrátor program Java nyelven íródott. A nyelvválasztás oka főleg a csapat eddigi tapasztalatán alapult, azonban döntéshez hozzájárult, hogy olyan programozási nyelvet válasszunk, mely futtatható Linux és Windows környezetekben is.

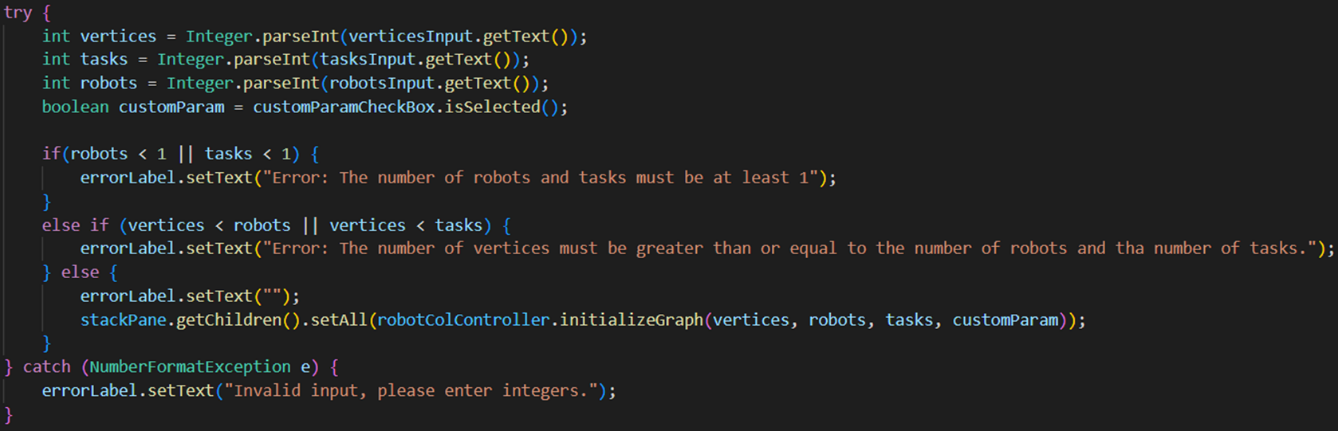
A program implementálja az egyetlen gráfot, melyhez optimális ütemterv állapítható meg, mely az útgráf, egyenlő hosszúságú úthosszokkal. A program a felhasználó által bevitt paraméterek alapján generál egy útgráfot, elhelyezi a gráf csúcsain a robotokat és az általuk elvégzendő feladatokat.

A generálás után a felhasználónak lehetősége van lépésenként végignézni az algoritmust futás közben, hogy melyik robot, mennyi idő alatt, milyen úton teljesíti a rászabott feladatsort.

A program a futás végén összegzi az eredményeket. Megjeleníteni csak a leghosszabb ütemtervet jeleníti meg, azonban készít egy log file-t az összes robot megtett útjáról annak érdekében, hogy az felkövethető legyen.

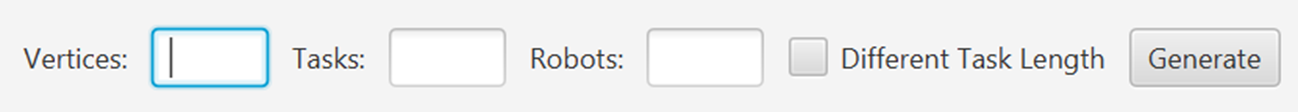
## Egyedi szituáció létrehozása

Azáltal, hogy a program bevonja a felhasználót a teszteset létrehozásába és felparaméterezésébe, bevonja a lehető legnagyobb hiba faktort, azt embert. Ennek okán az algoritmusnak már a legelején fel kell készülnie arra az esetre, hogyha a felhasználó hibás vagy hiányos adatot ad meg.



Négy különböző beviteli mező van, melyek mindegyike egy egész számot fogad el. Ezek a

* **Csúcsok mennyisége:** Az útgráfon elhelyezkedő csúcsok mennyisége, amin el fognak helyezkedni a robotok és feladatok.
* **Feladatok száma:** A gráfon elhelyezkedő feladatok mennyisége. Ennek a paraméternek feltétele, hogy nem lehet nagyobb érték, mint amennyi a csúcsok mennyisége.
* **Robotok száma:** A robotok mennyisége a gráfon. Ennek a paraméternek feltétele, hogy nem lehet nagyobb érték, mint amennyi a csúcsok mennyisége.
* **Változó feladat hosszúság:** Egy bool érték, mely azt határozza meg, hogy a generált feladatoknak a hossza fix legyen (1), vagy változó hosszúságú.

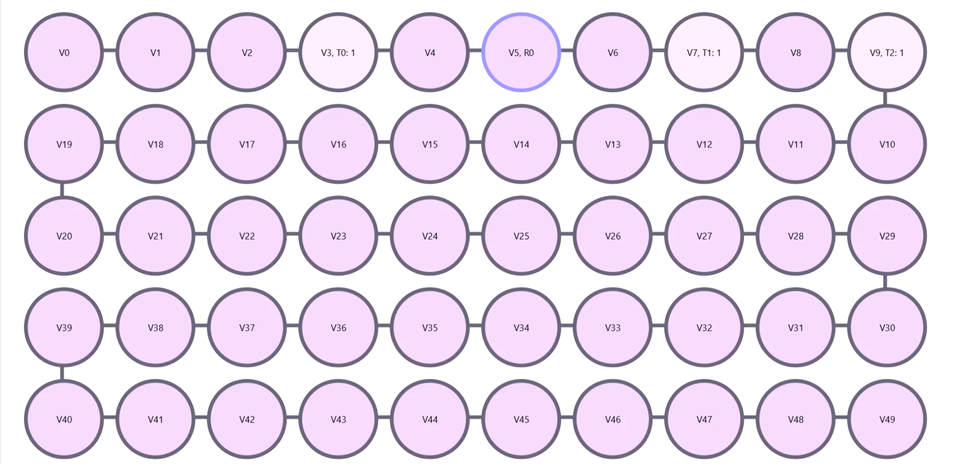


Az adatok megadása után a program legenerálja az útgráfot a megadott paraméterek alapján.

## Generált gráf

Az alábbi ábrán a bemenő paraméterek a következők voltak:

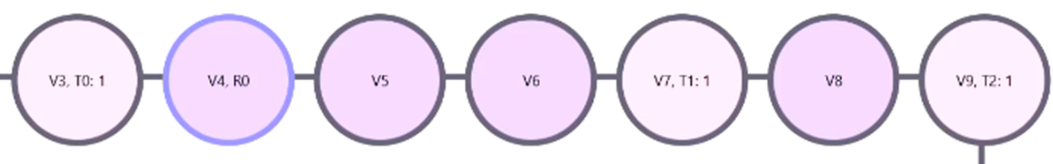
* 50 csúcs
* 3 feladat
* 1 robot
* Nem változó hosszúságú feladatok



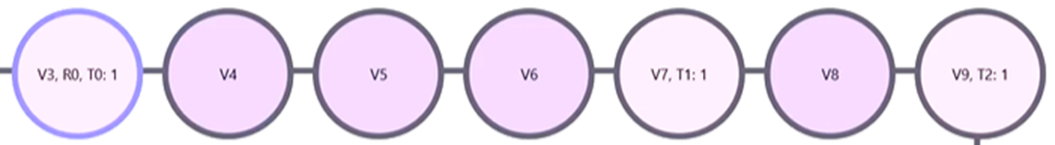
A csúcsok v0-tól v49-ig helyezkednek el. Azokat a csúcsokat, melyek nem tartalmaznak elvégzendő feladatot sötétebb, az elvégzendő feladatokat világosabb háttérszínnel jelöli a gráf. A csúcs, melynek különbözik a körvonala a többitől (V5) jelöli a robot kezdő pozícióját. A programot ezekután egy gomb megnyomásával tudjuk a következő iterációba átléptetni.

## Program futása

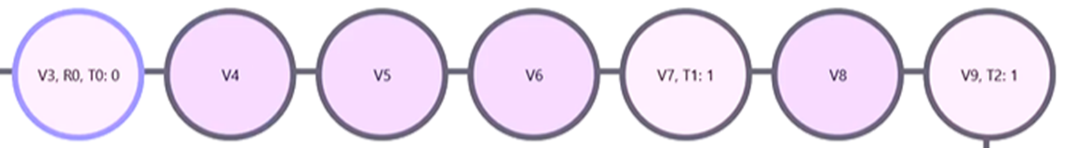
Az első iterációban a robot megindul. Annak érdekében, hogy el tudja dönteni, melyik irányba induljon az útvonalát a gráf adott részére korlátozza, ahol az adott robotnak kiosztott feladatok vannak. Ha a kezdőpontja közelebb van a legközelebbi feladat bal vagy jobb oldalához, akkor az ahhoz közeli irányba indul, figyelembe véve a többi robot ütemezését is az ütközések elkerülése érdekében.



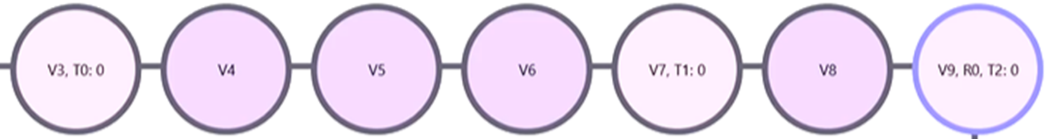
A második iterációban a robot eléri az első számára kiadott feladatot. Amikor a robot egy számára kijelölt feladatba belekezd, úgy azt neki kötelező is befejezni.



Ez a vizualizáció során abban mutatkozik meg, hogy a robot pontosan annyi iteráción keresztül tartózkodik pozíción, amennyi a feladat elvégzéséhez szükséges ().



A feladat elvégzését követően ismét elkezdhet mozogni és a következő feladat irányába menni. Jelen esetben ez a csúcson található.



Az utolsó feladat elvégzését követően a program megnézi, hogy melyik robotnak mennyi volt az összesített futásideje és a leghosszabbat ki is jelzi. Jelen esetben ez a , mint egyedüli robot, aki 11 ciklus alatt végzett a feladataival.

Ezzel egyidejűleg létrehoz egy log file-t is, ami lehetővé teszi a robotok lépéseinek, ütemtervének a monitorozását. Ezeket a log file-okat felhasználva készítettünk szimulációs kimutatásokat, ahol a bemeneti értékek változóak voltak.

# Szimulációk

A log fileokból kinyert adatokat egy python script-el vizualizáltuk, melynek a következők lettek az eredményei:

A képen szöveg, képernyőkép, Diagram, diagram látható

Automatikusan generált leírás

A diagram alapján megfigyelhető, hogy a robotok számának növelésével csökken az átlagos lépések száma, ami a hatékonyság javulására utal. Azonban bizonyos feladatok esetében (pl. 4-es vagy 5-ös feladat) a több robot használata nem mindig eredményez jelentős lépésszám-csökkenést, ami arra utalhat, hogy a robotok közötti ütemezés optimalizálása további fejlesztési lehetőségeket rejt.

A képen szöveg, képernyőkép, képernyő, szám látható

Automatikusan generált leírás

A fenti szórásdiagram a Steps és SumTaskLength közötti kapcsolatot ábrázolja különböző feladatok (Tasks) szerint. Az adatokból látszik, hogy a rövidebb feladatok (alacsonyabb SumTaskLength értékek) általában kevesebb lépéssel (Steps) teljesíthetők. Ugyanakkor a nagyobb SumTaskLength értékekhez gyakran több Steps tartozik, különösen bonyolultabb feladatoknál (pl. Tasks 9 és 10). A pontok mérete azt jelzi, hogy a robotok száma szintén befolyásolja az eredményt: több robot használata általában kevesebb lépést igényel az adott feladatok elvégzéséhez.

A képen szöveg, sor, diagram, Diagram látható

Automatikusan generált leírás

A fenti vonaldiagram a Steps és Tasks közötti kapcsolatot mutatja, ahol a vonalat körülvevő árnyékolás a változékonyságot jelöli (például a standard eltérést vagy más szórást). Az ábráról leolvasható, hogy a Steps értékek fel-le ingadoznak a feladatok számának növekedésével, ami arra utal, hogy a feladatok komplexitása és a hozzájuk rendelt robotok közötti ütemezési különbségek jelentősen befolyásolják a szükséges lépések számát. Az ingadozás csökkenése a magasabb Tasks értékeknél azt jelezheti, hogy az algoritmus hatékonyabban osztja el a lépéseket nagyobb feladatmennyiség esetén.

A képen szöveg, diagram, Tervrajz, Téglalap látható

Automatikusan generált leírás

A fenti „boxplot” diagram a Steps értékek eloszlását mutatja a különböző Tasks esetén. A diagram alapján látható, hogy a Steps értékek mediánja és szórása jelentősen változik a feladatok számától függően. Például a kisebb számú feladatoknál (pl. Task 1) a lépések száma szűk tartományban mozog, míg bizonyos feladatoknál (pl. Task 6) az eloszlás sokkal szélesebb, ami nagyobb változatosságot jelez az eredmények között.

A képen szöveg, képernyőkép, képernyő, Diagram látható

Automatikusan generált leírás

A fenti szórásdiagram a Steps és Robots közötti kapcsolatot vizsgálja. A diagram alapján látható, hogy a robotok számának növekedésével a lépések száma (Steps) jelentősen csökken. Egy robot esetén a Steps értékek nagyobb szórást és magasabb maximumot mutatnak, míg több robot esetén (pl. 4-6 robot) az értékek alacsonyabbak és sokkal koncentráltabbak. Ez azt mutatja, hogy a robotok számának növelése jelentős hatékonyságnövekedést eredményez az ütemezésben, mivel a feladatokat hatékonyabban lehet elosztani közöttük.

A képen szöveg, képernyőkép, diagram, Diagram látható

Automatikusan generált leírás

A fenti oszlopdiagram az átlagos Steps értékek változását mutatja a különböző Tasks számok esetében. Az ábrából látszik, hogy az átlagos lépésszám a feladatok számának növekedésével nem lineárisan változik. Például a 4-es és 6-os feladatoknál a Steps értékek kiemelkedően magasak, míg a 8-as és 10-es feladatoknál alacsonyabb átlagot figyelhetünk meg. Ez arra utal, hogy a feladatok komplexitása és a robotok mozgásának optimalizáltsága jelentős hatással van a teljesítményre.

# Konklúzió

A robotok ütemezésének problémája kritikus jelentőségű, különösen olyan környezetekben, ahol autonóm robotok hatékony és ütközésmentes működése elengedhetetlen. A tanulmány eredményei rámutattak, hogy a probléma általános esetben NP-teljes, de bizonyos gráfosztályoknál, mint például az útgráfok, hatékony algoritmusok alkalmazhatók. Az általunk kidolgozott program a cikk elméleti eredményeire alapozva képes a robotok ütemezésének vizualizálására és az optimális vagy közelítő megoldások meghatározására.

A programunk jelenlegi verziója elsősorban útgráfok kezelésére alkalmas. A jövőbeni fejlesztések egyik fő iránya lehet többféle gráf ábrázolásának és elemzésének lehetősége, például csillag vagy bipartit gráfoké. Ez nemcsak a program funkcionalitását növelné, hanem lehetővé tenné komplexebb problémák modellezését is. Továbbá a vizualizáció fejlesztésével és interaktív elemek beépítésével a felhasználók számára könnyebbé válna az ütemezési problémák megértése és megoldása.