UNIVERSITATEA DE STAT DIN MOLDOVA
Facultatea de Matematică și Informatică

Cu titlu de manuscris
C.Z.U: 004.728.5:519.85(043.3)

PAȘA TATIANA

ALGORITMI DE SOLUȚIONARE A PROBLEMELOR NELINIARE DE TRANSPORT

122.03 – MODELARE, METODE MATEMATICE, PRODUSE PROGRAM

Teză de doctor în informatică

Conducători științifici:
Ungureanu Valeriu,
dr. în șt. fiz.mat., conf. univ.
Paladi Florentin,
dr. hab. în șt. fiz. mat., prof. univ.

Autorul:
Pașa Tatiana

CHIȘINĂU, 2021

CUPRINS

ADNOTARE

Pașa Tatiana

“Algoritmi de soluționare a problemelor neliniare de transport”,
teză de doctor în informatică, Chișinău, anul 2021.

Structura tezei: Teza conține: introducere, trei capitole, concluzii generale și recomandări, bibliografie din 173 de titluri, 118 pagini text de bază, 10 figuri, 20 tabele. Rezultatele obținute sunt publicate în 19 lucrări științifice.

Cuvinte cheie: problemă de transport, problemă neliniară, rețea de transport, funcție de cost, funcție de producere și consum, graf, arbore de acoperire, algoritm genetic, mulțime admisibilă.

Scopul lucrării: constă în soluționarea unui șir de probleme neliniare de transport de dimensiuni mari cu funcții concave de cost.

Obiectivele cercetării: cercetarea problemelor de transport și a metodelor de soluționare a lor; elaborarea algoritmilor aproximativi pentru soluționarea problemelor neliniare de transport; elaborarea algoritmilor genetici pentru soluționarea în timp rezonabil a problemelor de transport de dimensiuni mari descrise de rețele cu una sau mai multe surse și una sau mai multe destinații cu funcții concave de cost; elaborarea algoritmilor genetici pentru soluționarea în timp rezonabil a problemelor de transport de dimensiuni mari descrise de rețele cu mai mulți indici și cu funcții concave de cost; testarea și estimarea timpului de execuție a algoritmilor propuși.

Noutatea și originalitatea științifică: constă în obținerea rezultatelor noi de ordin teoretico-aplicativ care completează cele cunoscute deja în literatura de specialitate în domeniul problemelor neliniare de transport. Sunt soluționate problemele neliniare de transport cu doi, patru și cinci indici aplicând reduceri consecutive a problemelor neliniare de transport la probleme liniare; este codificată problema neliniară de transport cu o sursă și o destitinație, problema neliniară de transport cu o sursă și câteva destinații, problema neliniară de transport cu câteva surse și câteva destinații, problema neliniară de transport cu 4 indici și cea cu 5 indici pentru a putea fi aplicați algoritmi genetici astfel încât la decodificare să fie obținute soluții admisibile; sunt descriși operatorii de încrucișare și mutație a algoritmilor genetici astfel încât la decodificare se obțin soluții admisibile; este demonstrat practic faptul că fiecare din algoritmii propuși generează o soluție admisibilă locală în timp rezonabil.

Rezultatul/rezultatele obținute care contribuie la soluționarea unei probleme științifice importante: consistă în *identificarea metodelor* de codificare a problemelor neliniare de transport fapt care *a condus la crearea* unor algoritmi genetici și a unor algoritmi bazați pe reducerea la probleme liniare *pentru implementrea ulterioară* în soluționarea problemelor aplicative.

Semnificația teoretică: este determinată de rezultatele obținute - elaborarea algoritmilor descriși de iterații polinomiale pentru soluționarea unor probleme neliniare de transport. Este demonstrat că algoritmii converg întotdeauna către o soluție optimă locală în timp rezonabil.

Valoarea aplicativă: constă în posibilitatea utilizării algoritmilor propuși pentru rezolvarea problemelor de transport reale și în adaptarea lor pentru o mulțime mai largă de probleme neliniare de transport.

Implementarea rezultatelor științifice: rezultatele obținute pot servi drept suport pentru cursuri opționale pentru studenții și masteranzi ce țin de soluționarea problemelor de optimizare neliniară. Algoritmii propuși permit soluționarea problemelor de aprovizionare a magazinelor cu produse, a întreprinderilor cu materii prime, a organizațiilor de construcție cu materiale de construcție etc..

АННОТАЦИЯ

Паша Татиана

«Алгоритмы решения нелинейных транспортных задач», диссертационная работа в информатике, Кишинэу, 2021 год.

Структура работы: Диссертация содержит: введение, три главы, заключение с рекомендациями, список цитируемой литературы состоящий из 173 наименований, 118 страниц основного текста, 10 рисунков, 20 таблиц. Полученные результаты были опубликованы в 19 научных работах.

Ключевые слова: транспортная задача, нелинейная задача, транспортная сеть, функция стоимости, функция производства и потребления, граф, дерево покрытия, генетический алгоритм, допустимое множество.

Цель исследования: состоит в том, чтобы решить ряд нелинейных транспортных задач больших размеров с вогнутыми функциями стоимости.

Задачи исследования: исследование транспортных задач и их методов решения; разработка приближенных алгоритмов решения нелинейных транспортных задач; разработка генетических алгоритмов для решения за разумное время крупномасштабных транспортных задач, описываемых сетями с одним или несколькими источниками и одним или несколькими пунктами назначения с вогнутыми функциями стоимости; разработка генетических алгоритмов для решения за разумное время крупномасштабных транспортных задач, описываемых сетями с несколькими индексами и вогнутыми функциями стоимости; тестирование и оценка времени выполнения предложенных алгоритмов.

Научная новизна и оригинальность: состоит в получении новых теоретико-прикладных результатов, дополняющих уже известные в специализированной литературе области нелинейных транспортных задач. Нелинейные транспортные задачи с двумя, четырьмя и пятью индексами решены путем применения последовательных сведений нелинейных транспортных задач к линейным задачам; была закодирована нелинейная транспортная задача с одним источником и одним стоком, нелинейная транспортная задача с одним источником и несколькими стоками, нелинейная транспортная задача с несколькими источниками и несколькими стоками, нелинейная транспортная задача с 4 и 5 индексами, так что, применяя генетические алгоритмы были получены допустимые решения во время декодирования; описаны операторы скрещивания и мутации генетических алгоритмов чтобы получить допустимые решения после декодирования; практически продемонстрировано что каждый из предложенных алгоритмов генерирует локально допустимое решение за разумное время.

Полученные результаты, способствующие решению важной научной проблемы: состоит в определение методов кодирования нелинейных транспортных задач, что привело к разработке генетических алгоритмов и алгоритмов, основанных на сведении к линейным задачам для последующего внедрения при решении прикладных задач.

Теоретическая значимость: определена полученными результатами, связанными с алгоритмами, описываемыми полиномиальными итерациями, предложенными для решения нелинейных транспортных задач. Доказано, что алгоритмы всегда сходятся к локальному оптимальному решению за разумное время.

Практическая применяемость: состоит в возможности использования предложенных алгоритмов при решении реальных транспортных задач и их адаптации для более широкого набора нелинейных транспортных задач.

Внедрение научных результатов: полученные результаты могут стать подспорьем при разработке некоторых факультативных курсов, связанных с решением задач нелинейной оптимизации, для студентов и магистрантов. Предложенные алгоритмы позволяют решить задачи обеспечения магазинов товарами, вывозимыми со складов, предприятий сырьём, строительных организаций строительными материаллами и т.д.

ANNOTATION

Pașa Tatiana

"Algorithms for solving nonlinear transportation problems"

a PhD degree in Informatics, Chișinău, 2021.

Thesis structure: The thesis contains an introduction, three chapters, general conclusions and recommendations, a bibliography of 173 titles, 118 pages of main text, 10 figures, 20 tables. The obtained results were published in 19 scientific papers.

Keywords: transportation problem, nonlinear problem, transportation network, cost function, production and consumption function, graph, spanning tree, algorithm, admissible set.

The goal of the thesis: is to study and solve large-scale nonlinear transportation problems with concave cost functions.

Research objectives: research of transport problems and methods of their solving; elaboration of approximate algorithms for solving non-linear transportation problems; elaboration of genetic algorithms for solving in a reasonable time the large-scale transportation problems described by networks with one or more sources and one or more destinations with concave cost functions; elaboration of genetic algorithms for solving in a reasonable time the large-scale transportation problems described by networks with several indices and concave cost functions; testing and estimating the execution time of the proposed algorithms.

The scientific novelty and originality: consists in new theoretical and applicative results that complete those known from specialized literature in the domain of nonlinear transportation problems. The non-linear transportation problems with two, four and five indices were solved through the application of consecutive reductions of the non-linear problems to linear ones; the non-linear problems with a single source and destination, with a single source and several destinations, with several sources and several destinations, with 4 indices and with 5 indices were codified so that genetic algorithms could be applied and admissible solutions are obtained through decoding; the crossover and mutations operators of the genetic algorithms were described so that admissible solution are obtained through decoding; it has been practically proved that each of the proposed algorithms generates a local admissible solution in reasonable time.

The obtained results which contribute to solve some important scientic problem: consists in indentification of methods of coding nonlinear transportation problems, which led to the creation of genetic algorithms and of algorithms based on reduction to a linear problem. These algorithms were then implemented to solve applicative problems.

The theoretical significance: of the research is determined by the obtained results related to the proposed algorithms with polynomial iterations for solving nonlinear transportation problems. It has been shown that algorithms always converge to a local optimal solution in a reasonable time.

The applicative value: of the paper consists in the possibility of using the proposed algorithms to solve real transportation problems and adapting them to a larger set of nonlinear transportation problems.

The implementation of the scientific results: the obtained results can serve as a support for some optional courses, related to solving nonlinear optimization problems, for bachelor or master students. The proposed algorithms allow solving the problem of supplying stores with products transported from warehouses, enterprises with raw materials, construction and civil engineering companies with construction materials, etc.

INTRODUCERE

Modelul problemei de transport a apărut din necesitatea soluționării unui grup de probleme economice în special a problemei ce vizează planificarea cheltuielilor de transport a produselor, fiind asigurată posibilitatea obținerii unor economii semnificative. În caz general, o problemă de transport este descrisă de transportarea produsului prin punctele rețelei din surse în destinații. Un aspect foarte important este costul de transport care în viața reală este descris de cele mai multe ori de o funcție neliniară, deoarece e dependent de mai mulți factori. Un loc aparte au și problemele ce țin de transportarea a mai multe tipuri de produse cu utilizarea a mai multe tipuri de transport.

Actualitatea și importanța temei de cercetare: Problema clasică de transport este o problemă de programare liniară, pentru a cărei soluționare a fost propusă metoda simplex (Dantzig, 1951), care constă în determinarea unui plan optim de transport pentru un produs omogen, de la surse identice cu scopul de a satisface necesitățile unor destinații identice, minimizând costul de transport produsului. Modelul problemei este utilizat pentru optimizarea aprovizionării întreprinderilor cu materie primă și materiale, dar și pentru aprovizionarea magazinelor cu produse acumulate în depozite. El are aplicație și dezvoltare în diferite domenii, cum ar fi proiectarea rețelelor de telecomunicații, a conductelor pentru apă, gaz, petrol, planificarea transportului rutier și a procesului de fabricație. Un șir de autori, precum: D. R. Fulkerson (1961), R. G. Busacker și P. J. Goven (1960), J. Edmonds și R. M. Karp (1972), N. Tomizawa (1971), M. Klein (1967), D. D. Sleator și R. E. Tarjan (1983), A. V. Goldberg și R. E. Tarjan (1987), (1989), (2014), P. T. Sokkalingam, R. K. Ahuja și J. B. Orlin, (2000), I-L.Wang și S-J. Lin (2009), L. Ciupală (2010), J. M. Davis și D. P. Williamson, (2012), P. Kovacs (2013), A. Sifaleras (2013), S. Ding (2014), M. Dawuni și K. F. Darkwah (2015), N. Baumstark et.al (2015), N. A. El-Sherbeny (2016), M. B. Cohen, A. Madry, P. Sankowski, și A. Vladu (2017), Md.A. Hakim și Md. R. Kabir (2017), A. M. P. Chandrasiri și D. M. Samarathunge (2017), R. A. Maher și F. A. Abdula (2017), J. Erickson, K. Fox și L. Lkhamsuren (2018), S. Abdi, F. Baroughi și B. Alizadeh (2018), au formulat diferite metode de soluționare a problemei de transport pe rețea pentru cazul în care funcțiile de cost sunt liniare, în cazul incertitudinii asupra costului și/sau a capacității asociate unui arc din rețea, au făcut o analiza teoretică și experimentală a algoritmilor de bază, au formulat concluzii și recomandări.

Un aspect important – funcțiile neliniare de cost complică soluționarea problemei de transport deoarece pentru rezolvare se cer a fi utilizate, spre exemplu, condițiile de optimalitate Kuhn-Tucker și multiplicatorii Lagrange, derivatele de ordinul întâi (metode gradient) și cele de

ordinul doi (matricea Hessian), dar și funcțiile de penalizare (Sun & Yuan, 2006), (Luenberger & Ye, 2008). Cazul funcțiilor concave de cost implică o serie de dificutăți suplimentare din cauza existenței mai multor puncte de extrem local, fapt ce complică obținerea soluției optime globale; aplicarea unor asemenea tehnici devine greoaie sau impracticabilă. Problemele de ultimul tip au fost cercetate de R. Horst și P. M. Pardalos (1995), R. Horst și H. Tuy (1996), Q. He, A. Shabbir și G. L. Nemhauser (2015), U. Klansec și M. Psunder (2010). Pentru un șir de metode de soluționare în (Gamețchi & Solomon, 2015) sunt expuse teoremele care demonstrează convergența acelor metode.

Dificultatea calculelor la soluționarea problemelor de transport menționate supra este condiționată de funcțiile concave de cost, dar și de dimensiunea problemelor studiate. Concavitatea funcției de cost apare natural grație faptului că costul unitar de transport poate fi mai mic pentru cantități mari de produse transportate, dar și grație existenței unor taxe fixe incluse în cost.

Pentru soluționarea unor astfel de modele complexe pot fi considerați algoritmii genetici, descriși pentru prima dată la Universitatea din Michigan de către J. Holland (1992). Faptul că algoritmii folosesc principii, proprietăți și noțiuni (populație, cromozomi, gene, selecție, încrucișare, mutație) din genetică a impus denumirea lor de „algoritmi genetici”. Algoritmi sunt euristici și stohastici, ceea ce înseamnă că soluția obținută nu întotdeauna este soluție optimă globală, ci doar se află în vecinătatea optimului global (adică este un pseudo optim). Algoritmii genetici sunt utilizați pentru soluționarea problemelor dificile. O analiză comparativă a algoritmilor genetici aplicați la rezolvarea problemelor de optimizare este prezentată în (Osaba, Carballedo, Diaz, Onieva, Iglesia, & Perallos, 2014), o trecere în revistă a diferiților algoritmi genetici fiind dată și în (Kudjo & Ocquaye, 2017).

Utilizarea algoritmilor genetici este recomandabilă pentru rezolvarea unor probleme de tipul celor menționate mai sus deoarece pentru aplicarea lor nu este necesară cunoașterea informației despre gradient sau Hessian, algoritmii sunt refractari la blocajele/ciclările în optime locale și algoritmii permit abordarea unor probleme de optimizare neliniară de dimensiuni mari. Deoarece în cazul problemelor de transport cu funcții concave de cost sunt frecvente situațiile evidențiate, e rațional ca la soluționarea lor să fie aplicați algoritmii genetici precum o fac, spre exemplu, autorii D.B.M.M. Fontes și J.F. Gonçalves (2007), A. Sadegheih și P. R. Drake (2009). Determinarea exactă a optimului global este dificilă, deoarece algoritmii genetici solicită utilizarea mai multor parametri și evaluarea funcției obiectiv de un număr mare de ori, ceea ce îngreuează aplicarea lor în scopul menționat.

Algoritmi genetici au fost propuși pentru soluționarea problemei de proiectare a rețelei de transport cu microcirculație ce permite utilizarea drumurilor adiacente pentru a diminua traficul intens de pe drumurile principale (Chen & Shi, 2012) și pentru soluționarea problemei de minimizare a timpului de reținere a avionului care are loc din cauza creșterii numărului de rute aeriene (Jiang, Xu, Zhang, & Luo, 2015), dar și pentru coordonarea mai multor linii de autobuze pe arterele urbane (Yang, Wang, Chen, Ding, & Li, 2015).

Algoritmul genetic descris în (Chen, Ni, Xu, Lv, & Wang, 2016) permite planificarea eficientă a stațiilor pentru trenurile de viteză înaltă. În (Zhang, Sun, & Liu, 2017) este abordată problema privind minimizarea costurilor pentru eficientizarea activității metroului fiind propusă soluționarea problemei prin aplicarea algoritmului genetic, iar în (Chen & Rilett, 2018) este dezvoltat un algoritm ce permite îmbunătățirea semnificativă atât a siguranței, cât și a eficienței coridoarelor de trafic pentru trecerile pe căile ferate. În (Bao, Gu, Di, & Zhang, 2018) autorii propun un algoritm genetic pentru problema optimizării rutelor de autobuz spre aeroporturi care au ca element de reper fiabilitatea timpului de transfer, aceasta fiind o problemă actuală, avându-se în vedere faptul că fiecare tinde să ajungă dintr-un punct în altul cât mai rapid, deci nu pot fi acceptate întârzieri la îmbarcare. În (Zhang, Mei, Liu, & Zheng, 2018) este studiată problema planificării suprafețelor de depozitare în porturile care fac parte din ruta de transportare a containerului cu scopul minimizării taxelor.

În teză sunt prezentate rezultatele studiului problemei neliniare de transport cu funcții concave de cost. Pentru soluționarea problemei sunt descriși algoritmi genetici care sunt testați în baza problemelor de transport pe rețele de diferite dimensiuni. În lucrare sunt formulate următoarele probleme generale:

Problema 1. Să se soluționeze problema neliniară de transport pe rețea cu funcții concave de cost cu una sau mai multe surse și una sau mai multe destinații prin care circulă un flux omogen de produs.

Problema 2. Să se soluționeze problema neliniară de transport cu funcții concave de cost descrisă de mai mulți indici: surse, destinații, puncte intermediare, tipuri de produse transportate, tipuri de transport utilizat.

Algoritmii de soluționare a Problemelor 1 și 2 sunt descriși în capitolele 2 și 3, respectiv.

Scopul lucrării: constă în studierea și soluționarea unui șir de probleme neliniare de transport de dimensiuni mari cu funcții concave de cost.

Obiectivele lucrării: pentru atingerea scopului sunt fixate următoarele *obiective*:

- cercetarea problemelor de transport cu și a metodelor de soluționare a lor;

- elaborarea algoritmilor aproximativi pentru soluționarea problemelor neliniare de transport;
- elaborarea algoritmilor genetici pentru soluționarea în timp rezonabil a problemelor de transport de dimensiuni mari descrise de rețele cu una sau mai multe surse și una sau mai multe destinații cu funcții concave de cost;
- elaborarea algoritmilor genetici pentru soluționarea în timp rezonabil a problemelor de transport de dimensiuni mari descrise de rețele cu mai mulți indici și funcții concave de cost;
- testarea și estimarea timpului de execuție a algoritmilor propuși.

Ipoteze de cercetare: în baza analizei literaturii de specialitate și a obiectivelor de cercetare, au fost dezvoltate următoarele ipoteze:

- problemele neliniare de transport cu funcții concave de cost formulate ca probleme de programare neliniară pot fi soluționate aplicând aproximarea succesivă la probleme liniare;
- aplicarea algoritmilor genetici permit soluționarea problemelor neliniare de transport de dimensiuni mari (câteva mii de necunoscute) în timp rezonabil;
- codificarea problemei neliniare de transport astfel încât să fie garantată obținerea soluțiilor admisibile la decodificarea cromozomilor este esențială;
- operatorii de încrucișare și mutație trebuie descriși astfel încât să asigure că prin aplicarea lor se vor obține cromozomi, cărora le sunt asociate soluții admisibile;
- utilizarea unui model elitist al algoritmilor genetici permite păstrarea soluțiilor admisibile ale problemei care pretind a fi soluții optime (pseudo optime) a problemei formulate.

Sinteza metodologiei de cercetare și justificarea metodelor de cercetare: cercetările științifice realizate în teză sunt bazate pe ultimile rezultate științifice din domeniul temei tezei de doctor descrise în literatura de specialitate și care se referă la noțiunile și metodele teoriei grafurilor, metodele de optimizare, teoria algoritmilor, teoria complexității, inteligența artificială și algoritmii genetici.

Noutatea științifică a rezultatelor obținute: constă în obținerea rezultatelor noi de ordin teoretico-aplicativ care completează cele cunoscute deja în literatura de specialitate în domeniul problemelor neliniare de transport și care sunt publicate în reviste recenzate. În baza rezultatelor teoretice sunt elaborați algoritmi eficienți care pot fi aplicați la soluționarea unui spectru larg de probleme practice. Noutatea științifică se exprimă prin:

- soluționarea problemei neliniare de transport pe rețea cu funcții concave de cost prin reducerea acesteia la o serie de probleme de programare liniară și demonstrarea convergenței algoritmului la un optim local „apropiat” de optimul global;

- codificarea problemelor neliniare de transport pe rețea cu una sau mai multe surse și una sau mai multe destinații ca urmare fiind elaborați algoritmi genetici de soluționare;
- soluționarea problemei neliniare de transport cu 4 indici și a problemei neliniare de transport pe rețea cu 5 indici prin reducerea acestora la o problemă de programare liniară;
- codificarea problemei neliniare de transport cu 4 indici și a problemei neliniare de transport pe rețea cu 5 indici ca urmare fiind elaborați algoritmi genetici de soluționare;
- pentru fiecare din algoritmii genetici elaborați au fost descrişi operatorii de încrucişare şi mutație care asigură că prin aplicarea lor se obțin cromozomi, cărora le sunt asociate soluții admisibile;
- demonstrarea convergenței către o soluție optimă locală (pseudo optimă) pentru fiecare dintre algoritmii elaborați.

Aprobarea rezultatelor științifice: rezultatele științifice de bază obținute și reflectate în lucrare, sunt prezentate la conferințe științifice naționale și internaționale, sunt publicate în reviste de specialitate din țară și de peste hotare:

a) Rezultatele științifice prezentate în secțiile conferințelor științifice:

1. Second Conference of the Mathematical Society of the Republic of Moldova dedicated to the 40th anniversary of the foundation of the Institute of Mathematics and Computer Science of ASM, 17 -19 August, 2004, Chişinău.
2. The international Conference on "Integral Equations and Problems of Applied Modelling", (IEMAP-2005), 20-25 June, 2006, Chișinău. http://www.uccm.md/biblioteca/IEMAP_Vol_2
3. A 20-a Conferință a SPSR, AFA "Henri Coandă", Departamentul de Științe Fundamentale și Management, 28-29 aprilie 2017, Brașov, România.
4. The Fourth Conference of Mathematical Society of the Republic of Moldova dedicated to the centenary of Vladimir Andrunachevici (1917 - 1997), 28 iunie – 2 iulie 2017, Institute of Mathematics and Computer Science, Academy of Sciences of Moldova, Chișinău.
5. International Conference on Information Technologies, Systems and Networks ITSN-2017, 17-18 October, 2017, Chisinau, Republic of Moldova.
6. The 50th anniversary of Computers, Informatics and Microelectronics Faculty & Electronics and Telecommunications Faculty, 9th International Conference "Microelectronics and Computer Science" & The 6th Conference of Physicistists of Moldova, October 19-21, 2017, Chisinau, Republic of Moldova.
7. A 21-a Conferință a Societății de Probabilități și Statistică din România, Academia de Studii Economice din București, 13-14 aprilie 2018, ASE, București, România.

8. International Conference on Mathematics, Informatics and Information technologies dedicated to the illustrious scientist Valentin Belousov, MITI 2018, 19-21 April, 2018, Bălți.
9. Moldova Wolfram Technology Conference 2018, June 9, USM, Chișinău.
10. CAIM 2018, The 26th Conference on Applied and Industrial Mathematics, 20th – 23th September, 2018, Technical University of Moldova, Chișinău, Moldova.
11. Conferința Internațională Modelare Matematică, Optimizare și Tehnologii informaționale, 12-16 noiembrie 2018, Chișinău.
12. MITRE, Moldova State University, June 24-26, 2019, Chișinău.
13. CAIM 2019, The 27th Conference on Applied and Industrial Mathematics, 19th – 20th September, 2019, "Valahia" University, Târgoviște, Romania.
14. IMCS - 55, The Fifth Conference of Mathematical Society of the Republic of Moldova, September 28 - October 1, 2019, Chișinău, Moldova.
15. International Conference on Applied and Pure Mathematics, 6th edition, ICAPM 2019, October 31 – November 3, Iași, România.
16. International Conference on Applied Mathematics and Numerical Methods, Third Edition, Craiova, October 29-31, 2020.
17. International Symposium ″Actual Problems of Mathematics and Informatics″ dedicated to the 90th Birthday of Professor Ion Valuță, November 27-28, 2020, TUM, Chișinău, Republic of Moldova.

b) Articole științifice publicate în reviste de specialitate din țară și de peste hotare, materialele (proceedings) conferințelor:

1. LOZOVANU, D., **PASHA, T.** An algorithm for solving the transport problem on network with concave cost functions of flow on edges. In: *CSJM, 2002, vol. 10,* no. 3(30), p. 341-347. ISSN 1561-4042.
2. **PAŞA, T.** Proprietățile soluționării optimale a problemei neliniare de transport pe rețea. În: *Analele Universității de Stat din Moldova, Seria "Științe fizico-matematice", 2004,* p. 134-140. ISSN 1811-2641.
3. **PAŞA, T.** Algoritmul determinării soluțiilor admisibile și a soluțiilor optime pentru problemele de transport pe rețea cu o singură sursă. In: *Analele Universității de Stat din Moldova, Seria "Științe fizico-matematice", 2005,* p. 188-190. ISSN 18811-2641.
4. **PAŞA, T.** Sinteza metodelor de soluționare a unor cazuri particulare ale problemei de transport pe rețea. În: *Studia Universitatis Moldaviae, Seria Științe Exacte, 2011, nr.7* (47), p. 53-59. Online ISSN 2345-1033.

5. **PAŞA, T.**, UNGUREANU, V. Asupra metodei potențialelor pentru problema de transport degenerată. In: *Studia Universitas Moldaviae, Seria Științe Exacte, 2017, nr. 2* (102), p. 30-36. ISSN 1857 – 2073.
6. **PAŞA, T.**, UNGUREANU, V. Solving the transportation problem with piecewise - linear concave cost function. In: *Review of AFA, The Scientific Informative Review, 2017, vol. XV no. 2 (34), p. 49 -56. SPSR, DOI: 10.19062/1842-9238.2017.15.2*. ISSN 1842-9238.
7. **PAŞA, T.** Problema fluxului maxim în rețele – analiza și sinteza algoritmilor de soluționare. În: *Studia Universitatis Moldaviae, Seria Științe exacte, 2017, nr. 7* (107), p. 150-158. ISSN 1857-2073, Online ISSN 2345-1033.
8. **PAŞA, T.**, UNGUREANU, V. Wolfram Mathematica as an environment for solving concave network transportation problem. In: *The Fourth Conference of Mathematical Society of the Republic of Moldova dedicated to the centenary of Vladimir Andrunachevici (1917 - 1997), 28 iunie - 2 iulie 2017*, p. 429 - 432. Chișinău: IMCS, AȘM. ISBN 978 - 9975 - 71 - 915 - 5.
9. **PAŞA, T.**, UNGUREANU, V. Non-Linear Concave Transportation Problem Solving and Implementation using Wolfram Language., In: *ITSN, 2017,* p. 30-39. Chișinău. ISBN 978-9975-3168-5-9.
10. **PAŞA, T.**, UNGUREANU, V. Applying sequential and parallel programming to solve a non-linear transport problem. In: *ICMCS, October 19 - 21, 2017*, p. 247-251. Cișinău, Republic of Moldova. ISBN 978-9975-4264-8-0.
11. **PAŞA, T.** The genetic algorithm for solving the non-linear transportation problem. In: *Review of the Air Force Academy, The Scientific Informative Review, 2018, Vol. XVI, nr. 2 (37), p. 37 - 44*. Online ISSN: 2069-4733, ISSN-L: 1842-9238.
12. **PAŞA, T.** Soluționarea problemei de transport pe rețea ca problemă a programării nelineare. În: *Studia Universitatis Moldaviae, Seria Științe Exacte, 2018, nr. 7* (117), p. 14-24. ISSN 1857 – 2073 Online ISSN 2345-1033.
13. **PAŞA, T.** Multi-index transport problem with non-linear cost functions. In: *Romai J., 2018, vol. 14,* no. 2, p. 129-137. Disponibil: https://rj.romai.ro/arhiva/2018/2/Pasa.pdf.
14. **PAŞA, T.**, The application of parallel programming in solving the non-linear transport problem, În: *Conferința Internațională Modelare Matematică, Optimizare și Tehnologii informaționale, 12-16 noiembrie 2018, Chișinău, p. 164-168*. ISBN 978-9975-62-421-3.
15. **PAŞA, T.**, UNGUREANU, V. Solving the non-linear 4-index transportation problem. In: *The Fifth Conference of Mathematical Society of the Republic of Moldova, 2019,* p. 221-224. Chișinău: Vladimir Andrunachievici Institute of Mathematics and Computer Science.

16. **PAȘA, T.** Solving non-linear multi-index transportation problems. In: *Romai J., 2019, vol. 15,* no. 2, p. 91-99. Disponibil: https://rj.romai.ro/arhiva/2019/2/Pasa.pdf.
17. **PAȘA, T.** Solving transportation problems with concave cost functions using genetic algorithms. In: *CSJM, 2020, vol. 28 no. 2* (83) *,* p.140-151. ISSN 1561-4042.
18. **PAȘA, T.** Algoritmi de soluționare a problemelor neliniare de transport cu mai mulți indici. În: *Studia Universitatis Moldaviae, Seria Științe Exacte, 2020, nr. 2* (132), p. 36-44. ISSN 1857 - 2073 Online ISSN 2345-1033.
19. **PAȘA, T.** Genetic algorithm for solving transportation problems on networks with one source and multiple sinks. In: *ITM Web Conf., ICAMNM 2020, Section: Applied Mathematics and Numerical Methods, nr. 34*, 11 pages. https://doi.org/10.1051/itmconf/20203402006, Online ISSN 2271-2097.

Sumarul compartimentelor tezei: Teza este structurată în trei capitole, în care sunt studiate o serie de cazuri particulare ale problemei neliniare de transport, fiind elaborați și testați algoritmii de soluționare a acestora. Lucrarea conține adnotări în limbile română, rusă și engleză, introducere, concluzii generale și recomandări, o listă bibliografică ce cuprinde 173 titluri, 12 anexe, declarația privind asumarea răspunderii și CV-ul autorului.

În **Introducere** este argumentată actualitatea și importanța temei de cercetare, sunt formulate scopul, obiectivele și ipotezele tezei. Este formulată problema științifică de importanță majoră fiind menționată importanța teoretică și valoarea aplicativă a lucrării, este efectuată o analiză a publicațiilor la tema tezei și o sinteză a conținutului tezei.

În **Capitolul I**, **Analiza situației în domeniul problemelor neliniare de transport**, format din 6 secțiuni, care poartă un caracter introductiv, este examinată situația actuală în domeniul de cercetare. Aici sunt enunțate rezultatele clasice și recente ce țin de domeniul de cercetare, și sunt descrise diferite clase de probleme de transport și algoritmi de soluționare, cum sunt: problema clasică de transport, problema de transport pe rețea cu funcții constante de cost, cu funcții liniare de cost, cu funcții neliniare diferențiabile, nediferențiabile și separabile.

Un rol important în soluționarea diferitor tipuri de probleme de transport au algoritmii genetici care sunt studiați sub aspectul posibilității de aplicare pentru obținerea soluțiilor în cazul problemelor complexe în ceea ce privește funcția obiectiv care descrie costul de transport, dar și problemele aplicative de dimensiuni mari.

Sunt grupate și analizate problemele de transport cu mai mulți indici pentru care funcția de cost este nu doar concavă și dependentă de sursele și de destinațiile rețelei, dar și de tipurile de produs transportat și de tipurile de transport utilizat. În acest context este propusă modalitatea de

soluționere prin reducerea unor astfel de probleme la serii de probleme de programare liniară, dar și prin aplicarea algoritmilor genetici în special pentru problemele practice de dimensiuni mari.

În **Capitolul II, Problema neliniară de transport cu funcții concave de cost,** format din 6 secțiuni, sunt formulate un șir de cazuri particulare ale problemei neliniare de transport pe rețea cu funcții concave de cost: *problema neliniară de transport pe rețea cu o sursă și o destinație*, *problema neliniară de transport pe rețea cu o sursă și o destinație pentru care se pune restricția că fluxul de produs trece prin toate arcele rețelei*, *problema neliniară de transport pe rețea cu o sursă și câteva destinații* și *problema neliniară de transport pe rețea cu câteva surse și câteva destinații*. Pentru fiecare din problemele formulate sunt descriși algoritmi de soluționare care generează soluție pseudo optimă în timp rezonabil.

Algoritmii AE1 și AE2 permit soluționarea problemei neliniare formulate prin reduceri succesive la o problemă a programării liniare pentru care sunt cunoscute metode de soluționare. Pentru acești algoritmi este formulată și demonstrată teorema de convergență către un optim local și cea de estimare a necesarului de memorie pentru implementarea algoritmului.

Un loc important îl ocupă algoritmii genetici AG1, AG2, AG3 și AG4 pentru care sunt aplicate diferite codificări în dependență de problema soluționată. Codificarea corectă a problemei permite obținerea unei soluții admisibile a problemei după decodificare. Sunt formulate teoremele de convergență către soluții $\mathcal{E}$ - optime pentru algoritmii AG1 și AG2 și teoremele de convergență către un optim local pentru algoritmii AG3 și AG4. Algoritmii elaborați sunt testați pentru probleme de diferite dimensiuni, fiind astfel demonstrate și practic atât convergența algoritmilor, cât și dependența timpului de execuție a algoritmului de parametrii rețelei de transport studiate.

În **Capitolul III, Problema neliniară de transport cu N indici,** format din 3 secțiuni, este studiată *problema de transport cu funcții concave de cost descrisă de 4 indici: surse, destinații, tipuri de produse transportate și tipuri de transport utilizate pentru transportare*. Problema este modelată matematic, sunt descrise proprietățile sale și elaborați doi algoritmi de soluționare. Algoritmul AME1 presupune reducerea problemei neliniare la o serie de probleme liniare. Pentru algoritmul genetic AMG1 problema este codificată astfel încât fiecărui cromozom i se asociază o soluție admisibilă a problemei după decodificare. Fiecare cromozom constă din 4 compartimente: lista de arce care descrie problema, lista de surse ordonate aleatoriu care descrie ordinea de deservire a surselor, lista de destinații ordonate aleatoriu care descrie ordinea de deservire a destinațiilor, lista tipurilor de produse și lista tipurilor de transport ordonate aleatoriu care descrie ordinea de deservire a produselor și destinațiilor respectiv. Sunt descriși operatorii de selecție, încrucișare și mutație aplicați asupra cromozomilor, astfel încât după aplicarea fiecăruia dintre acești operatori să se obțină un cromozom căruia i se poate asocia o soluție admisibilă.

Un interes deosebit prezintă studiul efectuat asupra *problemei de transport pe rețea cu funcții concave de cost care este descrisă de 5 indici: surse, destinații, tipuri de produse transportate, tipuri de transport utilizate pentru transportare și punctele intermediare ale rețelei*. Pentru a simplifica modelarea matematică a problemei dar și descrierea/implementarea algoritmilor de soluționare, problema este descrisă ca problemă cu *3 indici: arcele rețelei, tipuri de produse transportate și tipuri de transport utilizate pentru transportare*. Ca rezultat, este propus un algoritm AME2 care presupune reducerea problemei neliniare la o serie de probleme liniare. Este propus și algoritmul genetic AMG2 pentru care problema este codificată astfel încât fiecărui cromozom i se asociază o soluție admisibilă a problemei. Fiecare cromozom constă din 5 compartimente: lista arborilor de acoperire cu rădăcina în fiecare din sursele rețelei, lista de surse ordonate aleatoriu care descrie ordinea de deservire a surselor, lista de destinații ordonate aleatoriu care descrie ordinea de deservire a destinațiilor, lista tipurilor de produse și lista tipurilor de transport ordonate aleatoriu care descrie ordinea de deservire a produselor și transporturilor respectiv. Sunt descrișI operatorii de selecție, încrucișare și mutație care fiind aplicați asupra cromozomilor garantează obținerea unui cromozom căruia i se poate asocia o soluție admisibilă.

În compartimentul **Concluzii generale și recomandări** sunt expuse concluziile generale asupra rezultatelor obținute în cadrul tezei. Sunt punctate importanța și impactul acestor rezultate asupra dezvoltării domeniului dat. Sunt prezentate recomandările în formă de sugestii privind cercetările de perspectivă.

I. ANALIZA SITUAȚIEI ÎN DOMENIUL PROBLEMELOR DE TRANSPORT

O problemă de transport reprezintă o problemă de optimizare, care constă în determinarea unui plan optim de transport al unui produs omogen de la surse identice cu scopul de a satisface cerințele unor destinații identice minimizând costul de transport al acestui produs. Modelul problemei este utilizat pentru optimizarea aprovizionării întreprinderilor cu materie primă și materiale.

Analiza metodelor de soluționare a problemei neliniare de transport pe rețea ca problemă de programare neliniară a fost efectuată într-o serie de lucrări, precum sunt (Pașa, 2004), (Pașa, 2005), (Pașa, 2011), (Pașa, 2018a), iar a algoritmilor de soluționare a problemei fluxului maxim în rețea în lucrarea (Pașa, 2017c).

1.1 Problema de transport

Pentru prima dată problema de transport a fost formalizată de către matematicianul francez Gaspard Monge (1781). Problema a fost studiată de L. V. Kantorovich (1942) și de F. L. Hithcock (1941) care în 1941 a utilizat proprietățile problemei de transport doar la determinarea soluției inițiale. Mai târziu, T. C. Koopmans (1947) îmbunătățește metoda de rezolvare, iar Dj. Datzing (1951) realizează metoda simplex care, deși este una exponențială, s-a dovedit a fi una foarte eficientă și pe larg utilizată pentru soluționarea unei clase largi de probleme de programare liniară (Дантциг, 1966), (Гольштейн & Юдин, 1969).

Metoda potențialelor propusă de Kantorovici L. V. și Gavurin M. C. (1949) reprezintă o modificare a metodei simplex. Metoda potențialelor ține cont de specificul problemei de transport și conține un pas suplimentar de control al funcției la nemărginire pe mulțimea de soluții.

Problema de transport poate fi formulată în felul următor. Există m surse care pot livra produse omogene în cantitățile $a_i, i = \overline{1,m}$, și n destinații care solicită acest produs și au necesitățile $b_j, j = \overline{1,n}$. Se presupune că costul de transport a unei unități de produs de la sursa i la destinația j este c_{ij} u. c., unde $i = \overline{1,m}$, $j = \overline{1,n}$. Se cere determinarea unui plan optim de transport în scopul minimizării cheltuielilor totale, adică cantitățile x_{ij} de produs, unde $i = \overline{1,m}$, $j = \overline{1,n}$, care trebuie să fie transportate de la fiecare sursă la fiecare destinație, astfel încât costul total de transport să fie minim. Prin urmare, problema de transport constă în

determinarea unei soluții x^*, pentru care funcția $F(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$ atinge valoarea minimală.

Modelul matematic al problemei de transport poate fi scris astfel:

$$F(x) \to min$$

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i, & i = \overline{1,m} \\ \sum_{i=1}^{m} x_{ij} = b_j, & j = \overline{1,n} \\ x_{ij} \geq 0, & i = \overline{1,m}, j = \overline{1,n} \end{cases} \tag{1.1}$$

unde X descrisă de sistemul (1.1) este mulțimea de soluții admisibile.

Definiția 1.1. *Vom spune că problema de transport este echilibrată și admite soluții dacă satisface condițiile:*

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j\,, a_i \geq 0, b_j \geq 0, i = \overline{1,m}, j = \overline{1,n}$$

Problema de transport poate fi modelată matematic ca o problemă a programării liniare ce are ca scop minimizarea costului de transport.

În prezent, există pachete de programe de optimizare care conțin softuri performante pentru rezolvarea problemei de optimizare liniară. Un instrument bine cunoscut este Sistemul Matematica.

Problema liniară de transport poate fi soluționată utilizând metoda simplex (Dantzig, 1951). Însă, nu este de dorit să aplicăm această metodă din cauza caracteristicilor problemei și fiindcă matricea coeficienților din sistemul de restricții este dominată de zerouri, ceea ce conduce la o mulțime de operații care se vor îndeplini fără rost (în gol).

Se poate remarca faptul că într-o problemă de transport nu poate apărea decât varianta de optim finit existând întotdeauna soluții admisibile. Minimumul $-\infty$ nu este posibil, deoarece se cere minimizarea unei funcții liniare cu toți coeficienții pozitivi pe o mulțime de soluții cu toate componentele pozitive.

Problema de transport (1.1) echilibrată este un model de programare liniară cu $m + n$ restricții, $m * n$ variabile și care are întotdeauna o soluție admisibilă mărginită (Trandafir, 2004) de a_i, $i = \overline{1,m}$ și b_j, $j = \overline{1,n}$.

În Anexa 1 este prezentată o modalitate de tratare a soluției degenerate la soluționarea problemelor degenerate de transport (Pașa & Ungureanu, 2017b) și un exemplu ce descrie aplicarea acestei metode.

1.2 Problema fluxului de cost minim pe rețea de transport

La definirea noțiunilor de bază cu care se operează în această lucrare au fost consultate sursele (Ford & Fulkerson, 1956), (Берж, 1962), (Упсон, 1978), (Cormen, Leiserson, & Rivest, 2002) și (Corlat & Corlat, 2012), unde pot fi găsite și demonstrațiile teoremelor.

Definiția 1.2. *Numim rețea de transport un graf orientat* $G = (V, E)$ *fără cicluri, cu mulțimea de vârfuri* V, $|V| = n$ *și mulțimea de arce* E, $E \subset V \times V$, $|E| = m$ *și o funcție pozitivă* $u: E \to \mathbb{R}$ *- capacitatea arcului, care satisface următoarele proprietăți:*

1. *Există un vârf (sursă)* $v_s \in V$ *care nu are ascendenți;*
2. *Există un vârf (destinație)* $v_t \in V$ *care nu are descendenți;*

Definiția 1.3. *Numim flux într-o rețea funcția* $x: E \to \mathbb{R}$ *care posedă următoarele proprietăți:*

1. *Restricții de capacitate: pentru orice* $e \in E$ *are loc* $x(e) \leq u(e)$*;*
2. *Simetria: pentru orice* $(v_1, v_2) \in E$ *are loc* $x(v_1, v_2) = -x(v_2, v_1)$*;*
3. *Conservarea fluxului:* $\sum_{e \in E^+(v)} x(e) - \sum_{e \in E^-(v)} x(e) = 0$*, unde* $E^+(v)$ *este mulțimea de arce care intră în vârful* $v \in V$ *și* $E^-(v)$ *– mulțimea de arce care ies din vârful* $v \in V$*.*

Definiția 1.4. *Numim tăietură a unui graf* $G = (V, E)$ *o partiție* (S, T) *a mulțimii nodurilor, astfel încât* $v_s \in S$ *și* $v_t \in T$*, se notează* S/T*.*

Definiția 1.5. *Capacitatea tăieturii este suma capacităților tuturor arcelor cu vârful inițial în* S *și vârful final în* T*,* $(S, T) = \sum_{v \in S w \in T} u(v, w)$*.*

Definiția 1.6. (Goldberg & Tarjan, 1990) Fie graful $G = (V, E)$ cu fluxul f. Numim *graf rezidual* $G_f(V, E_f)$ graful cu mulțimea de vârfuri V care conține arcele cu capacități reziduale pozitive $E_f = \{(v, w) \in V \times V \,|\, u_f(v, w) > 0\}$.

Definiția 1.7. Numim *drum de creștere* drumul $p = (v_1, v_2, \dots, v_k)$ în graful rezidual cu $u_f(v_i, v_{i+1}) > 0$, $i = \overline{1, k-1}$, unde $v_1 = v_s$, $v_k = v_t$.

Definiția 1.8. *Mărimea maximă a fluxului net care intră în destinație* $|x| = \sum_{(v, v_t) \in E} x(v, v_t) - \sum_{(v_t, v) \in E} x(v_t, v)$ *se numește flux maxim în rețea.*

Teorema 1.1. (Fluxul Maxim - Tăietură Minimă) *(Ford & Fulkerson, 1956). Mărimea fluxului maxim, din sursă în destinație, într-un graf orientat $G = (V, E)$ este egală cu capacitatea tăieturii minime care separă sursa de destinație.*

Definiția 1.9. Numim *graf stratificat* toate drumurile minime între v_s și v_t în graful rezidual G_f, care formează subgraful $G_0 = (V_0, E_0)$ construit utilizând căutarea în lățime modificată (Dinic, 1970).

Definiția 1.10. *Numim capacitatea vârfului v, unde $v \in G_0(V_0, E_0)$ este vârf al grafului stratificat, mărimea $cap(v) = min\{\sum_{(u,v)\in E_0} u(u,v), \sum_{(v,w)\in E_0} u(v,w)\}$.*

Definiția 1.11. *Numim pseudoflux într-o rețea de transport funcția $f: E \to \mathbb{R}$ pentru care nu se respectă proprietatea $\sum_{e\in E^+(v)} f(e) = \sum_{e\in E^-(v)} f(e)$, unde $E^+(v)$ este mulțimea de arce ascendente în $v \in V$ și $E^-(v)$ – descendente din $v \in V$.*

Un flux admisibil este și pseudoflux. Deci, pseudoflux este fluxul pentru care cantitatea totală de flux care intră în vârf nu coincide cu cantitatea de flux care iese din vârf.

Definiția 1.12. Numim *capacitate reziduală a unui arc* funcția $u_f: E \to R$ pentru care $u_f(v,w) = u(v,w) - f(v,w)$ – fluxul care poate fi transportat suplimentar din v în w fără depășirea capacității $u(v,w)$.

Definiția 1.13. Numim *capacitate reziduală a unui drum p* cantitatea maximă de flux $u_r(p) = min\{u_r(v,w) | (v,w) \in p\}$ ce poate fi transportată de-a lungul său.

Cel puțin un arc pe drumul de creștere p are capacitatea reziduală $u_f(v,w) > 0$ până la îmbunătățire și $u_f(v,w) = 0$ după îmbunătățire, un astfel de arc este exclus din graful G_f și este numit *arc saturat*.

Definiția 1.14. *Numim flux saturat (Dinic, 1970) într-un graf orientat, fluxul x astfel încât fiecare drum $s \to t$ conține un arc cu capacitatea reziduală zero.*

Definiția 1.15. *Numim preflux (Goldberg & Tarjan, 1988) într-o rețea de transport funcția $x_f: E \to \mathbb{R}$ care respectă următoarele proprietăți:*

1. *pentru orice $e \in E$ are loc $x_f(e) \leq u(e)$;*
2. *pentru orice $(v_1, v_2) \in E$ are loc $x_f(v_1, v_2) = -x_f(v_2, v_1)$*
3. *pentru orice $v \in V$ are loc $\sum_{e\in E^+(v)} x_f(e) > \sum_{e\in E^-(v)} x_f(e)$, unde $E^+(v)$ este mulțimea de arce care intră în $v \in V$ și $E^-(v)$ – mulțimea de arce care ies din $v \in V$.*

Deci, preflux este fluxul pentru care cantitatea totală de flux care întră în vârf poate fi mai mare decât cantitatea de flux care iese din vârf.

Problema fluxului maxim în rețele de transport permite modelarea unor procese economice, cum ar fi proiectarea liniilor de electricitate, a rețelei de comunicații sau a rețelei de drumuri, a conductei de apă sau gaze sau optimizarea problemelor de producere și păstrare a mărfii în depozite, precum și optimizarea fluxurilor de pasageri.

Printre autorii care au descris algoritmi de determinare a fluxului maxim în rețele de transport sunt L. R. Ford și D. R. Fulkerson (1956), J. Edmonds și R. Karp (1972), E. Dinic (1970), A. Karzanov (1974), N. Megiddo (1974), Z. Galil și A. Naamad (1980), D. D. Sleator și R. E. Tarjan (1983), (1985), A. V. Goldberg și R. E. Tarjan (1988), G. L. Miller și J. Naor (1995).

Printre lucrările din ultimii ani, se pot enumera (Boycov & Kolmogorov, 2004), (Tarjan, Ward, Zhang, Zhou, & Mao, 2006), (Borradaile, Klein, Mozes, Nussbaum, & Wulff-Nilsen, 2011), (Borradaile & Klein, 2009), (Mehta, 2014), (Madry, 2016), (Mardy, 2013), (Cristiano, Kelner, Madry, Spielman, & Teng, 2011), (Sherman, 2013), (Kelner, Lee, Orecchia, & Sidford, 2014), (Ciupală, 2014), (Ciupală, 2016), (Șchiopu, 2016), (Holzhauser, Krumke, & Thielen, 2017).

Complexitatea unor algoritmi de soluționare a problemei fluxului maxim în rețea, autorii și anul descrierii acestor algoritmi sunt expuse în Anexa 2.

Problema fluxului de cost minim are un rol important în șirul problemelor de optimizare pe rețele și constă în transportarea unui flux de produs de la o mulțime de surse către o mulțime de destinații într-o rețea de transport cu restricții-capacități și funcții liniare de cost definite pe arcele rețelei. Problema are aplicație în diferite domenii, cum ar fi proiectarea rețelelor de telecomunicații, a conductelor pentru apă, gaz, petrol, planificarea transportului rutier, a aprovizionării cu produse și planificarea procesului de fabricație.

Definiția 1.16. *Două vârfuri ale unui graf orientat sunt conexe dacă există drum care le unește. Graful* $G = (V, E)$ *este conex dacă oricare două vârfuri sunt conexe. Dacă într-un graf orice pereche de noduri sunt conexe în ambele direcții cu drumuri orientate, vom spune că graful este tare conex.*

Se consideră graful conex orientat $G = (V, E)$, $|V| = n$, $|E| = m$, astfel încât fiecărui arc $e \in E$ îi este asociată o funcție $u: E \to \mathbb{R}$ numită capacitatea arcului și o funcție $c: E \to \mathbb{R}$ - costul de transport a uei unități de produs. Pe mulțimea V de vârfuri este definită o funcție reală mărginită de producere și consum $q: V \to \mathbb{R}$.

Problema fluxului de cost minim pe rețea de transport constă în aprovizionarea destinației de către sursă cu un flux de produs pe un drum cât mai ieftin posibil; deci, trebuie să se determine un flux $x^* \in X$ care minimizează funcția $F(x) = \sum_{e \in E} c(e)\, x(e)$, adică se cere soluționarea problemei liniare:

$$F(x) \to min,$$

$$\sum_{x \in E^+(v)} x(e) - \sum_{x \in E^-(v)} x(e) = q(v), \qquad v \in V,$$

$$0 \le x(e) \le u(e), \qquad e \in E,$$

unde $E^+(v) = \{(u,v) | (u,v) \in E\}$, $E^-(v) = \{(v,u) | (v,u) \in E\}$, X – mulțimea soluțiilor admisibile. Se presupune că toate datele sunt valori întregi și se dorește să se obțină ca soluție optimă o valoare întreagă fezabilă, iar $\sum_{v \in V} q(v) = 0$.

Teorema 1.2. *(Klein, 1967) $x \in X$ este flux de cost minim dacă și numai dacă nu există niciun ciclu orientat, astfel încât suma costurilor de-a lungul arcelor sale să fie o mărime negativă.*

În baza acestei teoreme se poate concluziona faptul că pentru ca un flux să fie optim trebuie de verificat dacă există un cost negativ într-un ciclu orientat. În caz că există un astfel de ciclu, fluxul este îmbunătățit transportând (împingând) un flux pozitiv de-a lungul arcelor ciclului ceea ce conduce la micșorarea costului total păstrând nemodificată mărimea fluxului.

Definiția 1.17. (Goldberg & Tarjan, 1990) *Funcția de cost* este o funcție $\pi: V \to R$ care descrie prețul unui vîrf v. *Funcția de reducere a costului* $c_\pi: E \to \mathbb{R}$ este definită de expresia $c_\pi(v,w) = c(v,w) - \pi(v) + \pi(w)$.

Definiția 1.18. *(Goldberg & Tarjan, 1990) Fie dat $\varepsilon \ge 0$, atunci un pseudoflux f se numește ε −optimal, $c_\pi(v,w) \ge -\varepsilon$, unde $(v,w) \in G_x$.*

Teorema 1.3. *(Goldberg & Tarjan, 1990). Fie costurile tuturor arcelor sunt întregi. Atunci pentru orice $0 \le \varepsilon \le 1/n$, un flux ε −optimal este flux optimal.*

În continuare, pentru descrierea complexității algoritmilor se folosesc notațiile: U – mărimea maximă a capacității arcelor, C – mărimea maximă a costului unei unități de produs transportat de-a lungul arcului.

Out of kilter este un algoritm descris de F. Fulkerson (1961). Utilizând procedura de marcare, fiecare arc care nu satisface condițiile de optim potrivit trebuie să fie adaptat. În cazul în care toate arcele se află în starea α, β, γ (descrise în (Fulkerson, 1961)), se spune că fluxul este unul admisibil și optim. Această stare se numește “in kilter”, în caz contrar “out of kilter”. Deci,

pentru a soluționa problema trebuie ca toate arcele să fie "in kilter", iar constanta kilter să fie egală cu un număr pozitiv. Algoritmul are complexitatea $O(|E|^3U)$.

Algoritmul descris de M. Iri (Iri, 1960) și R. G. Busacker cu P. J. Gowen (1960) are la bază algoritmul celor mai scurte drumuri, numit **Successive shortest path**. Primul algoritm polinomial de soluționare a problemei a fost descris de J. Edmonds și R. M. Karp (1970), (1972). Algoritmul se încheie în $O(|V|\ U\ Sp(|V|, |E|, |V|C))$ pași, unde Sp – timpul de determinare a celui mai scurt drum. Metoda celor mai scurte drumuri este prezentată și de N. Tomizawa, care în lucrarea sa (Tomizawa, 1971) descrie algoritmul pentru rețele cu câteva surse și câteva destinații.

M. Klein (1967) descrie algoritmul **Cycle canceling,** care pornește de la fluxul maxim, iar în rețeaua reziduală asociată se verifică existența ciclurilor orientate cu costuri negative. Dacă astfel de ciclu este depistat, atunci de-a lungul arcelor este transmis un flux pozitiv, astfel încât să fie saturat un arc care implică micșorarea costului, după care se construiește un nou graf rezidual. Dacă nu este depistat un ciclu orientat de cost negativ, atunci se spune că sa obținut fluxul de cost minim. Algoritmul necesită cel mult $2|E|CU$ iterații, iar timpul de execuție este de ordinul $O(|V||E|^2CU)$. În (Weintraub, 1974) autorul descrie modalitatea de soluționare a problemei fluxului de cost minim pentru cazul funcțiilor de cost convexe și al funcțiilor de cost liniare care la fel are la bază determinarea existenței ciclurilor de cost negativ.

În (Sokkalingam, Ahuja, & Orlin, 2000) autorii propun o metodă de identificare a ciclurilor de cost negativ care este realizată soluționând problema celor mai scurte drumuri cu arce de lungime nenegative. Algoritmul se execută în $O(|E|(|V| + |V|log|V|)\log(|V|U))$ timp.

Algoritmul descris de J. Edmond și R. M. Karp în (1972), numit și **Capacity-scaling algorithm**, prezintă o îmbunătățire a algoritmului celor mai scurte drumuri și este un algoritm slab polinomial. Algoritmul necesită determinarea celui mai scurt drum de $O(|E|logU)$, ceea ce presupune că timpul de execuție este $O((|E|logU)(|E| + |V|log|V|))$.

Pentru prima dată algoritmul **Cost-scaling** a fost descris de H. Rock (1980), ulterior de R. G. Bland și D. L. Jensen (1985). Acest algoritm are timpul de execuție $O(|V|\log(C)M(|V|, |E|, U))$, unde $M(|V|, |E|, U)$ reprezintă timpul necesar pentru determinarea fluxului maxim. A. V. Goldberg și R. E. Tarjan (1987) au perfecționat algoritmul aplicând ε-optimalitatea. Autorii descriu algoritmi care au limita superioară a complexității de $O(|V|^{5/3}|E|^{2/3}\log(|V|C))$, $O(|V|^3\log(|V|C)$, iar în cazul aplicării arborilor dinamici au complexitatea $O(|V||E|\log(|V|)\log(|V|C))$. În versiunea generică a algoritmului procedura de îmbunătățire ia $O(|V|^2|E|)$ timp, iar numărul total de faze a ε-scalării este de $O(\log(|V|)C)$ și, ca urmare rulează în timp slab-polinomial $O(|V|^2|E|\log(|V|C))$ (Goldberg & Tarjan, 1989).

A. V. Goldberg și R. E. Tarjan (1989) descriu algoritmul **Minimum-mean cycle-canceling,** care constă în determinarea ciclurilor de cost mediu minim și anularea lor; în cazul costurilor reale acesta se execută în $O(|V||E|^2\log(|V|))$ pași și necesită $O(|V|^2|E|^3\log(|V|))$ timp de execuție. În cazul costurilor întregi sunt necesari $O(|V||E|\log(|V|C))$ pași și $O(|V|^{\frac{3}{2}}|E|^2\min\{\log^2(|V|C), \sqrt{|V|}\log(|V|C), \sqrt{|V|}|E|\log(|V|)\})$ timp.

O îmbunătățire a algoritmului precedent a fost propus de A. V. Goldberg și R. E. Tarjan (1989). Algoritmul **Cancel and tighten** are la bază doi pași: 1) sunt determinate și anulate ciclurile admisibile până când graful admisibil este aciclic, ceea ce implică anularea a cel mult $|E|$ cicluri; 2) este modificat vectorul potențialelor π, astfel încât ε este micșorat până la cel mult $(1 - 1/|V|)$ din valoarea sa precedentă. În cazul costurilor întregi sunt executate $O(|V|\log(|V|C))$ iterații, cu implementarea arborilor dinamici în $O(|E|\log(|V|))$ timp pe iterație, ceea ce implică un $O(|V||E|\log(|V|)\log(|V|C))$ timp total de determinare a fluxului de cost minim.

Algoritmul **Double scaling** (Ahuja, Goldberg, Orlin, & Tarjan, 1988) se bazează pe o combinație a câtorva tehnici, cum sunt capacity-scaling (Edmonds & Karp, 1972), excess-scaling (Orlin, 1988), cost-scaling (Goldberg & Tarjan, 1988), dar și utilizarea arborilor dinamici (Sleator & Tarjan, 1983), care conduce la rularea sa în $O(|V||E|\,\text{loglog}\,U\log(|V|C))$ timp.

Algoritmul simplex pe rețea (Dantzig, 1951), (Dantzig, 1963) poate fi privit ca un caz particular a algoritmului *cycle calceling,* iar soluția, atât cea inițială, cât și cea optimă, este descrisă de un arbore de acoperire. Îmbunătățirea soluției constă în adăugarea unui arc nou care conduce la apariția unui ciclu de cost negativ care apoi este anulat prin saturarea altui arc și excluderea lui din componența arborelui obținând un nou arbore, procedură ce se repetă până când nu mai pot fi găsite astfel de arce care ar putea fi adăugate. Degenerarea (Cunningham, 1976) – apariția unui ciclu de capacitate reziduală zero (costul fluxului nu poate fi micșorat) este evitată prin garantarea că poate fi transportată o cantitate de flux pozitiv din orice vârf pe un drum al arborelui de acoperire. Algoritmul este studiat atât din punctul de vedere al eficienței de implementare (Grigoriadis, 1986), (Lobel, 1996), (Armstrong & Jin, 1997), cât și posibilitatea dezvoltării unor cazuri particulare (Tarjan R. E., 1991), (Orlin, Plotkin, & Tardos, 1993), (Wang & Lin, 2009).

Fiind o problemă actuală, cercetătorii studiază modalități de îmbunătățire atât teoretică în ceea ce privește complexitatea de execuție a algoritmilor, cât și modalități de implementare practică, în special în cazul soluționării problemelor de diferite dimensiuni (Ciupală, 2010), (Ghiyasvand, 2012a), (Kiraly & Kovas, 2012), (Ghiyasvand, 2012b), (Goldberg, Kaplan, Hed, & Tarjan, 2015), (Dawuni & Darkwah, 2015), (Ahmed, Khan, Ahmed, & Uddin, 2016), (El-Sherbeny, 2016), (Cohen, Madry, Sankowski, & Vladu), (Chandrasiri & Samarathunge, 2017),

(Goldberg, Hed, Kaplan, & Tarjan, 2017), (Maher & Abdula, 2017), (Erickson, Fox, & Lkhamsuren, 2018).

Un șir de lucrări descriu posibilitatea de soluționare a cazurilor particulare vizând problema fluxului maxim folosind algoritmul simplex pe rețea (Geranis, Paparizzos, & Sifaleras, 2009), (Ebrahimnejad & Nasseri, 2012), (Mizuno, Sukegawa, & Deza, 2014).

Algoritmii 2- și 3-aproximativi sunt propuși în (Cheriyan & Laekhaukit, 2013), (Rostami & Ebrahimnejad, 2014), iar în (Davis & Williamson, 2012), (Couetoux, Davis, & Williamson, 2015) autorii descriu un algoritm 3/2-aproximativ care soluționează problema de determinare a costului minim pentru o mulțime de arce, astfel încât fiecare componentă conexă are cel puțin k vârfuri.

Mai mulți autori (Han, Peng, & Wang, 2013), (Ding, 2014), (Guo, Wang, & Zhou, 2015), (Abdi, Baroughi, & Alizadeh, 2018) propun metode pentru soluționarea problemei în cazul incertitudinii asupra mărimii costului și/sau a capacității asociate unui arc din rețea. Analiza teoretică și experimentală a algoritmilor de bază este realizat în așa lucrări precum (Kovacs, 2013), (Sifaleras, 2013), autorii formulând unele concluzii și recomandări.

1.3 Problema neliniară de transport

O problemă de transport presupune că este necesară minimizarea (costului, pierderilor) sau maximizarea (profitului) unei funcții, astfel încât să fie satisfăcute un șir de condiții descrise de restricțiile de egalități și/sau inegalități. Deci, se cere soluționarea unei probleme de programare liniară sau neliniară în dependență de tipul funcției obiectiv și de restricțiile egalități și/sau inegalități.

Neliniaritatea problemei provine din rigurozitatea descrierii fenomenelor economice, fapt care duce la apariția unor dificultăți în determinarea soluției optime. Deși nu este formulată o metodă eficientă de soluționare a problemei de programare neliniară sub forma sa generală, se pun condiții funcției obiectiv și/sau restricțiilor astfel soluționându-se unele cazuri particulare.

În cele ce urmează ne propunem studierea metodelor de soluționare a problemei neliniare de transport pe rețea și cercetarea cazurilor când sunt impuse restricții de egalități și de ne negativitate, fapt ce ar corespunde condiției de existență a fluxului în rețeaua de transport studiată.

1.3.1 Formularea problemei neliniare. Noțiuni de bază

Se consideră problema de transport pe rețea descrisă de graful conex aciclic $G = (V, E)$, $|V| = n$, $|E| = m$. Pe mulțimea de vârfuri este definită funcția reală de producere și consum

$q: V \to R$. Pe mulțimea de arce sunt definite funcțiile neliniare de cost $\varphi_e(x_e)$. Problema neliniară de transport pe rețea constă în determinarea unui flux x^* care minimizează funcția $F(x) = \sum_{e \in E} \varphi_e(x_e)$, adică se cere soluționarea problemei neliniare:

$$F(x) \to min \tag{1.2}$$

$$\sum_{e \in E^+(v)} x(e) - \sum_{e \in E^-(v)} x(e) = q(v) \tag{1.3}$$

$$x(e) \geq 0, \ \ e \in E \tag{1.4}$$

unde X este mulțimea care satisface sistemul de ecuații (1.3) și restricțiile de pozitivitate (1.4), $E^-(v) = \{(v,u) | (v,u) \in E\}$, $E^+(v) = \{(u,v) | (u,v) \in E\}$, iar sistemul de restricții (1.3) – (1.4) definește mulțimea de soluții admisibile X ale problemei formulate.

Definiția 1.19. *Mulțimea fezabilă (mulțimea soluțiilor admisibile) X a problemei neliniare de transport este descrisă de restricțiile (1.3) și (1.4).*

Definiția 1.20. *Punctul $x^* \in \mathbb{R}^m$ este punct de minim (maxim) global dacă și numai dacă $x^* \in X$ și $f(x^*) \leq f(x)$ $[f(x^*) \geq f(x)]$ oricare ar fi $x \in X$, și este punct de minim (maxim) strict global dacă și numai dacă $x^* \in X$ și $f(x^*) < f(x)$ $[f(x^*) > f(x)]$ oricare ar fi $x \in X \setminus \{x^*\}$.*

Definiția 1.21. *Punctul $x^* \in \mathbb{R}^m$ este punct de minim (maxim) local dacă și numai dacă $x^* \in X$ și există o vecinătate $\mho$ a lui x^* astfel încât $f(x^*) \leq f(x)$ $[f(x^*) \geq f(x)]$, oricare ar fi $x \in X \cap \mho$, și este punct de minim (maxim) strict local dacă și numai dacă $x^* \in X$ și există o vecinătate $\mho$ a lui x^* astfel încât $f(x^*) < f(x)$ $[f(x^*) > f(x)]$, oricare ar fi $x \in (X \cap \mho) \setminus \{x^*\}$.*

Punctele de minim și maxim ale funcției se mai numesc și puncte de extrem.

Definiția 1.22. *Gradientul unei funcții f în raport cu o variabilă vectorială $x = (x_1, x_2, \dots, x_m)$, notat ∇f, este un câmp vectorial ale cărui componente sunt derivatele parțiale ale funcției, deci: $\nabla f = (\frac{\partial f}{x_1}, \frac{\partial f}{x_2}, \dots, \frac{\partial f}{x_m})$.*

Dificultatea soluționării problemelor neliniare, în comparație cu a celor liniare, constă în faptul că diferențierea dintre extremul local și cel global este imposibilă și că soluționarea unei probleme neliniare astfel încât să fie obținută o soluție globală este foarte complicată. Deseori, este complicată și obținerea unei soluții locale.

Teorema 1.4. *(Teorema Weiestrass - de existență a punctului de minim). Dacă mulțimea fezabilă $X \subset \mathbb{R}^m$ este compactă (închisă și mărginită) și $f: X \to \mathbb{R}$ este continuă, atunci există un punct de minim global pentru problema (1.2) - (1.4).*

Teorema 1.5. *(Condiții necesare de existență a punctului de minim local) Dacă x^* este punct de minim local și există toate derivatele parțiale ale funcției f în x^*, atunci $\nabla f(x^*) = 0$, adică $\frac{\partial f}{\partial x_j} = 0$ pentru $x_j = x_j^*,\ j = \overline{1,m}$.*

Teorema 1.6. *(Condiții suficienta de extrem) Fie x^* un punct staționar al funcției f (adică $\nabla f(x^*) = 0$). Fie funcția f diferențiabilă într-o vecinătate a lui x^* și de două ori diferențiabilă în x^* (adică există $\nabla^2 f$ în x^*). Dacă $\nabla^2 f(x^*)$ este pozitiv semidefinită, atunci x^* este punct de minim local al lui f.*

Teorema 1.7. *(Condiții suficienta de extrem) Fie x^* un punct staționar al funcției f (adică $\nabla f(x^*) = 0$). Fie funcția f diferențiabilă într-o vecinătate a lui x^* și de două ori diferențiabilă în x^* (adică există $\nabla^2 f$ în x^*). Dacă $\nabla^2 f(x^*)$ este pozitiv definită, atunci x^* este punct de minim strict local al lui f.*

Având un punct staționar x^* al funcției f pentru a determina dacă acesta este punct de minim local, poate fi inclusiv verificat dacă matricea Hessiană $H(x^*)$ (formată din derivatele parțiale de ordinul al doilea ale funcției f calculate în x^*) este pozitiv definită. Iar pentru a verifica dacă o matrice este pozitiv definită, poate fi utilizat criteriul lui Sylvester, conform căruia o matrice simetrică este pozitiv definită dacă și numai dacă toți minorii ei principali sunt pozitivi.

1.3.2 Algoritmi de soluționare a problemei cu funcții diferențiabile

În caz general, determinarea soluției unei probleme neliniare începe cu o soluție inițială admisibilă. Soluția optimă se obține iterativ, astfel încât la pasul k avem: $x_{k+1} = x_k + \lambda_k d_k$, $k = 1,2, \ldots$, unde $d_k \in \mathbb{R}^m$ este o direcție de deplasare din punctul x_k, iar λ_k este un scalar care descrie lungimea pasului de deplasare. Metodele de optimizare diferă prin modalitatea de calcul al mărimilor λ_k și d_k, dar se va ține cont și de faptul că la fiecare iterație se va obține o soluție x_k admisibilă, iar valoarea funcției obiectiv se va micșora. Vor fi impuse niște condiții de optimalitate care ne vor garanta că x_k este punct de minim al funcției. Numărul de pași sunt restricționați de timpul de execuție a algoritmului, de complexitatea calculelor sau de capacitățile tehnicii utilizate.

În cazul în care, la fiecare iterație k vectorul d_k indică direcția de descreștere a funcției $f(x)$ în punctul x_k, adică se determină un punct nou $x_{k+1} = x_k + \lambda_k d_k$ astfel încât $f(x_{k+1}) < f(x_k)$, se vorbește despre o metodă de descreștere a funcției $f(x)$.

Conform **metodei gradient** (Cauchy, 1847)**,** vectorul direcției de descreștere d_k se va lua egal cu antigradientul, $d_k = -\nabla f(x_k)$ cu $x_{k+1} = x_k - \lambda_k \nabla f(x_k)$, unde $k = 0,1,2, \ldots$, iar x_0 este un punct inițial cunoscut. Se consideră $k = 0$, se alege un punct inițial x_0 și se calculează $\nabla f(x_0)$

și o constantă $\varepsilon > 0$, care este utilizată pentru stoparea procesului de calcul. Dacă $\|\nabla f(x_k)\| = \sqrt{\sum_{i=1}^{m} \left(\frac{\partial f}{\partial x_i}\right)^2} < \varepsilon$, atunci STOP cu x_k soluție optimă. Altfel, se consideră $d_k = -\nabla f(x_k)$, se determină λ_k soluționând problema de minimizare a funcției $f(x_k + \lambda_k d_k)$ pentru $\lambda_k > 0$. Se trece la iterația $k+1$ considerând $x_{k+1} = x_k + \lambda_k d_k$.

O astfel de abordare funcționează eficient la primele iterații ale minimizării, dar în apropierea punctului staționar, din cauza pașilor mici în direcția de descreștere, devine ineficientă.

Definiția 1.23. *Fie M o matrice simetrică și pozitiv definită. Vectorii* $d_0, d_1, \dots, d_k \in \mathbb{R}^m$ *se numesc vectori reciproc conjugați în raport cu matricea M de ordinul m, dacă toți vectorii sunt diferiți de zero și* $(d_j M d_i) = 0$ *pentru* $i \neq j$, $i, j = \overline{1, m}$.

Metoda gradienților conjugați este potrivită, deoarece orice funcție poate fi destul de bine aproximată cu o funcție pătratică în vecinătatea punctului de minim, ceea ce garantează obținerea punctului minim într-un număr finit de pași. Deoarece după fiecare set de calcule care constă din n iterații are loc pasul descris de cea mai rapidă descreștere, metoda permite o apropiere foarte bună de minim, fapt ce implică posibilitatea soluționării problemelor de dimensiuni mari.
Inițial, pentru $k = 0$ se alege un punct inițial x_0, se calculează $\nabla f(x_0)$ și se consideră $d_0 = -\nabla f(x_0)$. Se alege o constantă $\varepsilon > 0$, care este utilizată pentru stoparea procesului de calcul.

Pasul de bază. Dacă $\|\nabla f(x_k)\| < \varepsilon$, atunci STOP cu x_k soluție optimă, altfel se trece la pasul 1.

Pasul 1. Se determină λ_k soluționând problema de minimizare a funcției $f(x_k + \lambda_k d_k)$ pentru $\lambda_k > 0$. Se trece la iterația $k+1$ considerând $x_{k+1} = x_k + \lambda_k d_k$. Dacă $k < n$, atunci se trece la pasul 2, altfel – la pasul 3.

Pasul 2. Se consideră $d_{k+1} = -\nabla f(x_{k+1}) + \mu_k d_k$, unde $\mu_k = \frac{\|\nabla f(x_{k+1})\|^2}{\|\nabla f(x_k)\|^2}$ și se trece la pasul 1 substituind k cu $k+1$.

Pasul 3. Se consideră $x_0 = x_n$, $d_0 = -\nabla f(x_n)$ și $k = 0$ și se trece la pasul de bază.

Metoda Newton utilizează și calculul derivatelor parțiale de ordinul doi, ceea ce presupune că matricea Hessiană $H(x_k)$ este nedegenerată pentru $\forall x_k$. Metoda converge numai dacă punctul inițial este destul de aproape de punctul optim, fapt ce constituie neajunsul metodei. Inițial se consideră $k = 0$, se alege un punct inițial x_0 și se consideră aproximarea funcției bazată pe dezvoltarea Taylor în vecinătatea punctului x_0. Este soluționată problema: $\min f_k(x)$, cu $f_k(x) = \nabla f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$. Pentru a determina x_{k+1} se verifică condiția necesară de optimalitate, adică dacă derivatele parțiale ale funcției $f_k(x)$ sunt nule, deci

$\nabla f(x_k) + H(x_k)(x - x_k) = 0$. Astfel, $x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k)$, $k = 0,1,2, \ldots$ este formula recurentă pentru punctele generate. Dacă $\nabla f(x^*) = 0$ și $H(x^*)$ este pozitiv definită atunci x^* este punct de minim local.

Metoda Newton are mai multe modificații care permit soluționarea problemei indiferent de care este punctul inițial ales x_0. Printre aceste modificații se poate numi **metoda Newton cu pas reglabil** sau **metoda Newton cu aproximația matricei Hessiane**.

Deoarece minimumul funcției obiectiv pentru probleme de optimizare cu restricții se poate atinge și pe frontiera mulțimii X ce descrie mulțimea soluțiilor admisibile ale problemei, pe lângă metodele clasice de soluționare a problemelor de optimizare fără restricții pot fi aplicate și unele metode adaptate la particularitățile descrierii mulțimii de restricții, dar și la tipul funcției obiectiv care se cere a fi minimizată.

Metoda multiplicatorilor Lagrange permite soluționarea problemelor de optimizare, cu restricții de egalități și inegalități, prin transformarea lor în probleme de optimizare fără restricții și aplicarea algoritmilor optimizării fără restricții care garantează convergența către un punct staționar. Aceste puncte staționare pot fi de minim local sau global, de maxim local sau global dar și puncte de inflexiune.

Condiția de regularitate Slater. *Există* $x' \in \mathbb{R}^m$ *cu proprietatea* $h_i(x') < 0,\ i = \overline{1,n}$ *și* $x'_j > 0,\ j = \overline{1,m}$ *ceea ce presupune că mulțimea soluțiilor admisibile are interiorul relativ nevid.*

Condițiile de optimalitate locală sunt descrise de următoarea teoremă, iar punctele care le satisfac sunt puncte staționare ale problemei.

Teorema 1.8. *Fie funcțiile* $f(x)$, $h_1(x)$, ..., $h_n(x)$ *continuu diferențiabile în vecinătatea punctului* $x^* \in \mathbb{R}^m$. *Dacă* x^**este punct de minim local, atunci există* $\lambda_0^*, \lambda_1^*, \ \ldots\ , \lambda_n^*$ *concomitent nenule, astfel încât:*

$$\nabla_x L(x^*, \lambda^*) = \lambda_0{}^* \nabla f(x^*) + \sum_{i=1}^{n} \lambda_i{}^* \nabla h_i(x^*) = 0$$

în punctul x^*. *Dacă totodată gradienții* $\nabla h_i(x^*)$, $i = \overline{1,n}$ *sunt liniar independenți, atunci* $\lambda_0 \neq 0$.

Nu întotdeauna punctele staționare sunt soluțiile problemei, iar pentru a le selecta pe cele optime este aplicată teorema ce urmează care conține condițiile suficiente de optimalitate.

Teorema 1.9. *Fie funcțiile* $f(x)$, $h_1(x)$, ..., $h_n(x)$ *de două ori diferențiabile în punctul staționar* $x^* \in \mathbb{R}^m$ *și fie:*

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\partial^2 L(x^*,\lambda^*)}{\partial x_i \partial x_j}dx_i dx_j > 0,$$

pentru orice creșteri nenule dx_i și dx_j, astfel încât:

$$\sum_{j=1}^{m}\frac{\partial h_i(x^*,\lambda^*)}{\partial x_j}dx_j = 0, \qquad i=\overline{1,m}.$$

Atunci x^* este punct de minim local strict al problemei inițiale sau a problemei de minimizare a funcției lui Lagrange.

Din cauza dificultăților de aplicare a metodei Lagrange, aceasta nu este utilizată pe larg la soluționarea problemelor de optimizare. Deseori sunt recomandate metode numerice aproximative care garantează o soluție din vecinătatea punctului optim.

Teorema 1.10. *(Teorema Kuhn-Tucker) (Kuhn & Tucker, 1951). Punctul x^* este soluție optimă a problemei de programare convexă $\min F(x)$, $h_i(x) \le 0$, $i=\overline{1,n}$, dacă și numai dacă există un punct $\lambda^* \ge 0$, astfel încât (x^*,λ^*) este punct șa al funcției Lagrange, adică $L(x^*,\lambda) \le L(x^*,\lambda^*) \le L(x,\lambda^*)$.*

Dacă funcțiile $f(x)$, $h_1(x)$, ..., $h_n(x)$ sunt diferențiabile, atunci se poate demonstra că condițiile teoremei precedente sunt echivalente cu condițiile de optimalitate Kuhn-Tacker:

$\nabla_{x_j} L \ge 0,\ x_j^* \nabla_{x_j} L = 0,\ x_j^* \ge 0,\ j=\overline{1,m}$ și $\nabla_{\lambda_i} L \le 0,\ \lambda_i^* \nabla_{\lambda_i} L = 0,\ \lambda_i^* \ge 0,\ i=\overline{1,n}.$

În cazul în care funcția obiectiv este una convexă și domeniul de definiție a soluțiilor admisibile la fel este convex, avem situația când minimul local coincide cu minimul global. Pentru astfel de probleme și sunt obținute un șir de rezultate.

Aplicarea metodelor gradient clasice în cazul existenței restricțiilor presupune deplasarea de-a lungul celei mai rapide descreșteri și poate duce în puncte inadmisibile. În scopul evitării unor astfel de situații, de către J. B. Rosen a fost propusă **metoda gradientului proiectat** (Rosen, 1960), conform căreia antigradientul este proiectat astfel încât punctul obținut să fie unul admisibil, iar funcția să ia valori mai mici (să se minimizeze).

Definiția 1.24. *Proiecția punctului $a=(a_1,a_2,\dots,a_n)$ pe mulțimea X se numește punctul $\Pi_X(a)$, care este cel mai aproape de a dintre toate punctele din X.*

Deci, de fapt efectuarea proiectării unui punct pe o mulțime presupune soluționarea problemei de proiectare care nu este una simplă. În cazul unor probleme complicate se poate recurge la unele modificații ale metodei gradientului proiectat, când mulțimea soluțiilor admisibile este aproximată cu o mulțime de poliedre.

Definiția 1.25. *Matricea pătratică P de ordinul n se numește matrice de proiectare, dacă $P = P'$ și $P \cdot P = P$, unde P' este transpusa matricei P.*

Într-un punct admisibil x, direcția celei mai rapide descreșteri este vectorul antigradient, dar deoarece mișcarea pe astfel de direcție poate duce la un punct în afara domeniului soluțiilor admisibile, vectorul este proiectat. Deci, trebuie proiectat vectorul $-\nabla f(x)$, astfel încât să se poată mișca de-a lungul direcției $d = -P\nabla f(x)$, unde P este matricea respectivă de proiectare.

Metoda funcțiilor de penalizare (Sun & Yuan, 2006) (Luenberger & Ye, 2008) constă în reducerea problemei de optimizare cu restricții la un șir de probleme de optimizare fără restricții de tipul $\min_{x \in X} \Phi(f, h, g, r)$, unde r este parametrul de penalizare a funcției de penalizare Φ. Funcția de penalizare descrisă este minimizată de un șir de valori r crescătoare, care forțează mulțimea de minime obținute să tindă către optimul problemei cu restricții, iar complexitatea de soluționare a problemei este de același ordin ca cea a problemei inițiale.

Pot fi descrise metode care modifică funcția obiectiv începând cu puncte situate atât în interiorul domeniului de admisibilitate, cât și în exteriorul lui, dar pot fi descrise și metode ce combină aceste două tipuri.

În cazul în care problema (1.2) – (1.4) este formulată ca problemă cu restricții inegalități, ea poate fi soluționată utilizând **metoda barieră,** care constă în transformarea problemei cu restricții în problemă de minimizare fără restricții, introducând funcții barieră ce nu permit ieșirea punctului considerat din domeniul valorilor admisibile.

1.3.3 Algoritmi de soluționare a problemei cu funcții ne diferențiabile

Descrierea proceselor economice duce deseori la soluționarea unor probleme neliniare de transport care presupun aplicarea metodelor de optimizare ne diferențiabile. Astfel de metode permit soluționarea problemelor fără a fi impuse schimbări în formularea problemei sau fără ajustarea funcției obiectiv care ar duce la descrierea procesului economic cu unele abateri de la cel real.

La bază sunt metodele gradient clasice (descrise în secțiunea precedentă). În acest scop este introdusă noțiunea de gradient generalizat (subgradient) pentru funcții ne diferențiabile, care reprezentă un echivalent al noțiunii de gradient în punctele în care, de fapt, gradientul nu există, dar și o descriere a procedurii de descreștere și adaptare a pasului pentru direcția nou-obținută la fiecare pas al metodei.

Fie $f(x)$ funcție convexă definită pe E^m, X^* mulțimea de minime (care poate fi și vidă), $x^* \in X^*$ punct de minim, $\inf f(x) = f^*$, $d_f(x) -$ subgradientul funcției în punctul x.

Definiția 1.26. *(Шор, 2008). Gradientul generalizat (subgradientul)* $d_f(x_0)$ *al funcției* f *în punctul* x_0 *este un vector* $d_f(x_0)$ *pentru care* $f(x) - f(x_0) \geq \left(d_f(x_0), x - x_0\right)$ *pentru orice* $x \in E^m$.

Deci, rezultă că dacă $f(x) < f(x_0)$, atunci $\left(d_f(x_0), x - x_0\right) > 0$, ceea ce presupune că în punctul x_0 antisubgradientul formează un unghi ascuțit cu orice direcție aleatoare, dusă din x_0 în direcția x cu o valoare mai mică a lui $f(x)$. De unde avem că dacă X^* nu este mulțime vidă, iar $x_0 \notin X^*$, atunci la deplasarea din x_0 în direcția $d_f(x_0)$ cu un pas destul de mic distanța până la X^* se micșorează.

Definiția 1.27. *(Шор, 2008). Metodă a gradientului generalizat vom numi procedura de construire a șirului* $\{x_k\}_{k=0}^{\infty}$ *de minimizare, unde* x_0 *este o mărime inițială, iar* x_k *se construiesc după următoarea formulă recurentă:* $x_{k+1} = x_k - \lambda_{k+1} \frac{d_f(x_k)}{\|d_f(x_k)\|}$, $k = 0,1,2, \ldots$, *unde* $d_f(x_k)$ *este un subgradient aliator a funcției* $f(x)$ *în punctul* x_k, *iar* λ_{k+1} *este multiplicatorul pasului.*

Metoda astfel definită este descrisă de procedura de alegere a mărimilor pasului λ_k care satisfac condițiile: $\lambda_k > 0;\ \lambda_k \to 0\ pentru\ k \to \infty;\ \sum_{k=1}^{\infty} \lambda_k = +\infty$ și pot fi utilizate următoarele metode de alegere a pasului:

1. $\lambda_{k+1} = \frac{\lambda_0}{k+1},\ \lambda_0 > 0,\ k = 0, 1, 2, \ldots;$
2. $\lambda_{k+1} = \frac{\lambda_k}{k+1},\ \lambda_0 > 0,\ k = 0, 1, 2, \ldots;$
3. $\lambda_{k+1} = \frac{\lambda_k + \lambda_{k-1}}{2},\ \lambda_0 > 0,\ \lambda_1 = \frac{\lambda_0}{2},\ k = 1, 2, \ldots;$

În dependență de modalitatea de alegere a șirului $\{x_k\}$ poate fi aplicată:

- **metoda gradientului generalizat cu pas constant,** conform căreia pentru $\lambda > 0$ pasul este $\lambda_k(x_k) = \frac{\lambda}{\|d_f(x_k)\|}$ și se obține $x_{k+1} = x_k - \lambda \frac{d_f(x_k)}{\|d_f(x_k)\|}$, $k = 0,1,2, \ldots .;$
- **metoda gradientului generalizat cu șirul convergent al multiplicatorilor pasului fără normarea gradientului** cu $x_{k+1} = x_k - \lambda_{k+1} d_f(x_k)$, $k = 0, 1, 2, \ldots;$
- **metoda gradientului generalizat cu șirul convergent al multiplicatorilor pasului fără normarea gradientului și cu posibilitatea de revenire în punctul inițial**: $x_{k+1} = \begin{cases} x_k - \lambda_{k+1} d_f(x_k), & \lambda_{k+1} \|d_f(x_k)\| \leq 0, \\ x_0 & în\ caz\ contar, \end{cases}$ $k = 1, 2, \ldots$ cu $c > 0$ o constantă aleatoare;
- **metoda gradientului generalizat cu pas constant, iar apoi cu pas micșorat de două ori**: pentru $\sigma \geq 2$, $x_{k+1} = x_k - \lambda_{k+1} \frac{d_f(x_k)}{\|d_f(x_k)\|}$, unde $\lambda_{k+1} = h_0 \cdot 2^{-[(k+1)/N]}$ cu h_0 suficient de mare și $N \geq 3\sigma^2 + 1$.

Pentru toate metodele menționate în (Gametchi & Solomon, 2015), (Шор, 2008) sunt formulate teoremele respective care demonstrează convergența metodelor și sunt specificate condițiile în care se recomandă aplicarea fiecăreia. În dependență de problema formulată este aleasă o metodă sau alta, ceea ce conduce la o soluție cât mai aproape de cea exactă.

1.3.4 Soluționarea problemei neliniare cu funcții separabile

Deseori, la modelarea unei probleme economice reale, deși modelul depinde de mai multe variabile, funcțiile care sunt aplicate depind doar de o singură variabilă. De exemplu, când vorbim despre problema neliniară de transport pe rețea se poate menționa faptul că sunt funcții de cost asociate fiecărui arc din rețea; respectiv fiecare depinde de mărimea fluxului transportat pe acel arc: funcția obiectiv este sumă de funcții neliniare dependente de o singură variabilă, iar funcțiile restricții sunt sume de funcții liniare dependente de o singură variabilă. În acest context, un caz special cercetat în continuare este soluționarea problemei neliniare de transport ca problemă a programării neliniare separabile.

Definiția 1.28. *O funcție f cu m necunoscute se numește separabilă dacă poate fi scrisă ca sumă de m funcții, fiecare de câte o variabilă, astfel:* $f(x_1, \dots, x_m) = f_1(x_1) + \cdots + f_m(x_m)$.

Definiția 1.29. *O problemă de programare neliniară se numește separabilă dacă funcția obiectiv și toate restricțiile sunt considerate funcții separabile, ceea ce înseamnă că este formulată problema:*

$$F(x) = \sum_{i=1}^{m} f_i(x_i) \to min,$$

$$\sum_{i=1}^{m} h_{ji}(x_i) \le b_j, j = \overline{1,n}, \qquad x_i \ge 0, i = \overline{1,m}.$$

Deoarece funcția obiectiv $F(x) = \sum_{e \in E} \varphi_e(x_e)$ a problemei neliniare de transport formulată mai sus este o sumă de funcții fiecare dependentă de o singură variabilă, iar funcțiile restricții sunt funcții liniare, rezultă că sunt satisfăcute condițiile care descriu o problemă a programării separabile. Printre primii care au formulat o metodă de soluționare a problemei de programare separabilă a fost C. E. Miller (1963), dar a fost studiată și în (Li & Yu, 1999). Pentru soluționarea problemei separabile cu funcții convexe de cost au fost propuși algoritmi polinomiali în (Minoux, 1986), (Vegh, 2012).

Condiții suficiente pentru un optim global, al problemei programării matematice la minim ar fi ca funcția obiectiv să fie convexă și mulțimea soluțiilor admisibile să fie o mulțime convexă.

În caz că funcția obiectiv este concavă, iar mulțimea soluțiilor admisibile este mulțime convexă, putem vorbi doar despre un minim local.

Metodele de soluționare a unor astfel de probleme presupun soluționarea unei sau a câtorva probleme ale programării liniare generate de problema inițială. Ele au la bază aproximarea funcției neliniare cu o funcție liniară sau secvențial-liniară și aplicarea metodei simplex sau simplex-modificată, în dependență de caz.

Schema algoritmului ce presupune aproximarea cu o funcție liniară.

Pasul inițial. Se determină o soluție admisibilă inițială $x^0 = (x_1^0,\ x_2^0, \dots, x_m^0)$ a problemei neliniare care reprezintă o soluție a sistemului de restricții.

Pasul 1. Dacă soluția obținută este optimă, atunci STOP. Pentru funcția obiectiv neliniară se determină gradientul $\nabla f(x^k) = (\frac{\partial f(x^k)}{x_1}, \frac{\partial f(x^k)}{x_2}, \dots, \frac{\partial f(x^k)}{x_m})$, unde k este numărul iterației curente.

Pasul 2. Se construiește funcția $F(z) = \frac{\partial f(x^k)}{x_1} z_1 + \frac{\partial f(x^k)}{x_2} z_2 + \dots + \frac{\partial f(x^k)}{x_m} z_m$ care este una liniară, apoi, pentru aceleași restricții, se determină valoarea minimală $z^k = (z_1^k, z_2^k, \dots, z_m^k)$ a funcției.

Pasul 3. Pentru soluția z^k a problemei liniare se determină soluția problemei inițiale neliniare $x^{k+1} = x^k + \lambda^k(z^k - x^k)$, unde λ^k este pasul de calcul $0 \leq \lambda^k \leq 1$ care poate fi luat aliator sau poate fi ales astfel încât $f(x^{k+1})$ să fie minimal. Se trece la pasul 1.

Această metodă, deși permite obținerea soluției optime pentru problema neliniară separabilă, depinde foarte mult de soluția inițială aleasă la începutul algoritmului.

Pentru obținerea unor soluții mai bune, se propune aproximarea funcției obiectiv neliniare separabile cu o funcție secvențial – liniară. Problema liniară cu funcția obiectiv secvențial – liniară este soluționată aplicând metoda simplex (sau o modificare a acesteia) în baza căreia se obține soluția pentru problema inițială neliniară.

Este cunoscut faptul că orice funcție convexă poate fi aproximată cu un careva grad de acuratețe cu o funcție secvențial-liniară, fapt care implică aproximarea unei probleme a programării neliniare, cu o careva acuratețe, cu o problemă a programării liniare. Programarea separabilă poate fi aplicabilă și în cazul funcțiilor non-convexe, dar trebuie de menționat că în așa caz nu ne este garantat optimul global.

1.4 Algoritmi genetici de soluționare a problemei de transport

Un interes deosebit prezintă problemele de transport care descriu situațiile reale din economia modernă, fapt ce implică modele complexe pentru a căror soluționare un șir de autori propun algoritmi genetici. Acești algoritmi au fost descriși pentru prima dată la Universitatea din Michigan sub conducerea lui J. Holland (1992). Utilizarea de către algoritmi a principilor, proprietăților și a noțiunilor din genetică (populație, cromozomi, gene, selecție, încrucișare, mutație) a condiționat denumirea de algoritmi genetici. Astfel de algoritmi sunt euristici și stohastici (întâmplători), ceea ce presupune că soluția obținută nu întotdeauna este cea optimă, dar se află într-o vecinătate cu soluția optimă. În caz general, astfel de algoritmi sunt algoritmi polinomiali.

Un domeniu interesant de aplicare a algoritmilor genetici este medicina. Utilizarea lor cu aplicații promițătoare în diferite domenii (radiologie, cardiologie, pulmonologie, endocrinologie, ș.a.) permite screeningul bolii, diagnosticul și planificarea tratamentului (Ghaheri, Shoor, Naderan, & Haseini, 2015).

Un algoritm genetic original este cel propus în (Thomas & Kovoor, 2018), care permite soluționarea problemei de parcare prin implementarea unui sistem inteligent de parcare și altul propus în (Liu, Lin, & Hsueh, 2019), care permite soluționarea problemei de transport și logistică a produselor. O imagine de ansamblu a diferitor tehnici de algoritmi evolutivi utilizați pentru soluționarea problemelor de optimizare multimodală a fost oferită în (Cansas, 2015).

O analiză comparativă a algoritmilor genetici aplicați în probleme de optimizare este prezentată în (Osaba, Carballedo, Diaz, Onieva, Iglesia, & Perallos, 2014), precum și o trecere în revistă a diferitor algoritmi genetici este dată în (Kudjo & Ocquaye, 2017).

Chiar dacă nu se cunosc algoritmi genetici care să reprezinte o soluție omniprezentă la toate problemele de optimizare, utilizarea lor este recomandată, deoarece pentru aplicare nu este necesară cunoașterea informației despre gradient sau hessian; în plus, ei sunt rezistenți la blocaje într-un optim local fapt ce le permite să iasă din minime locale chiar dacă structura restricțiilor care descriu domeniul soluțiilor admisibile este destul de complexă și pot oferi mai mult de o soluție optimă. Dat fiind faptul că se poate face față unui număr mare de variabile fără o creștere semnificativă a timpului de calcul, algoritmii genetici sunt reușiți pentru soluționarea problemelor de optimizare neliniară de dimensiuni mari (cu multe variabile). Algoritmii genetici sunt potriviți pentru calculatoarele paralele, deoarece generarea populației și calcularea valorilor funcțiilor obiectiv pot fi realizate în paralel. Numărul mare de soluții admisibile care permit algoritmilor genetici să acopere mai bine spațiul de căutare are ca rezultat și o convergență lentă. Acest fapt

care implică o alegere foarte atentă a dimensiunii populației generate de algoritm pentru a nu reține obținerea soluției optime.

Deoarece problema de transport cu funcții concave de cost aparține clasei de probleme NP-dificile, aceasta înseamnă că nu există un algoritm eficient pentru determinarea unei soluții într-un timp rezonabil. Problema este prea complicată pentru a fi rezolvată exact pentru rețele de transport mari, motiv din care ne bazăm pe un algoritm euristic. Se poate spune că în acest caz poate fi aplicat un algoritm genetic de soluționare, după cum o fac și autorii D.B.M.M. Fontes și J. F. Gonçalves (2007), A. Sadegheih și P. R. Drake (2009), O. M. Surakhi, M. Qatawneh, și H. A. Ofeishat (2017). Aplicarea algoritmilor genetici la soluționarea problemelor de transport complexe este studiată și în teze de doctorat după cum o fac și autorii (Басова, 2004), (Дубравина, 2005). Deoarece determinarea exactă a optimului global este dificilă se impune ca necesară utilizarea mai multor parametri, precum și evaluarea funcției obiectiv de un număr mare de ori, ceea ce îngreunează aplicarea lor.

Algoritmii genetici au la bază teorema schemelor (Paiu, 1998).

Definiția 1.30. *O schemă H este o machetă (șablon, template) care descrie o submulțime de cromozomi (indivizi din populație) cu secțiuni genetice similare.*

Se poate vorbi despre două proprietăți ale schemelor:

- Ordinul unei scheme H, notat $o(H)$ – numărul de poziții fixe prezente în șablon;
- Lungimea de definire a unei scheme H, notată $\delta(H)$ – distanța dintre prima și ultima poziție specificată în șir.

Teorema 1.11. *(Teorema schemelor)* (Paiu, 1998) *Schemele de lungime de definire scurtă, de ordin redus, aflate peste medie, obțin un număr de încercări exponențial crescător în generațiile următoare.*

Un prim pas în utilizarea unui algoritm genetic este codificarea corectă a problemei, adică descrierea elementelor (genelor) cromozomilor care reprezintă codificarea unei soluții admisibile a problemei. În cele mai dese cazuri este utilizată codificarea binară, dar poate fi și numerică, simbolică sau caracterială, în dependență de problema formulată. O populație constă din cromozomi care, de fapt, reprezintă codificarea unei mulțimi de soluții admisibile.

La crearea unei populații trebuie să fie întrunite următoarele condiții:

- cromozomii utilizați sunt de lungime constantă;
- numărul cromozomilor din populație este constant;

– fiecare populație $P(i+1)$ se obține din urmașii populației $P(i)$ sau din părinții și urmașii populației $P(i)$.

Definiția 1.31. *Valoarea funcției obiectiv totală a populației, notată $F_T(x)$, este suma valorilor funcțiilor obiectiv pentru toate soluțiile obținute prin decodificarea cromozomilor unei populații.*

Este preferat modelul elitist a algoritmului genetic fapt care implică păstrarea cromozomilor promițători din populație pentru populația următoare. Aplicarea algoritmului genetic trebuie să respecte un șir de reguli când are loc trecerea de la o populație la alta. În acest scop se aplică *selecția* cromozomilor care permite depistarea acelor cromozomi a unei populații care pot participă la crearea populației noi.

Astfel, se poate aplica una dintre metodele de selecție:

- Probabilitatea de selecție a unui cromozom depinde de valoarea performanței acestuia;
- Cromozomii sunt sortați în ordinea creșterii valorii funcției obiectiv și probabilitatea de selecție depinde de poziția lor în șir;
- Pentru doi cromozomi aleși aliator se calculează valoarea funcției obiectiv și este ales cel pentru care valoarea este mai mică (în probleme de minimizare).

În ceea ce urmează, este aplicată selecția care presupune că cromozomii din populație sunt sortați în ordine crescătoare a funcției obiectiv în soluția asociată fiecărui cromozom din populație. În populația nouă sunt transferați cromozomii din prima jumătate a populației, deci cei pentru care costul de transport este minim.

Cromozomii selectați și transferați în populația nouă joacă rolul cromozomilor-părinți prin *încrucișarea* cărora sunt obținuți cromozomii-urmași. Cromozomii-părinți sunt luați la rând câte doi cărora li se aplică o *tăietură* aleatorie ceea ce înseamnă că genele unui cromozom sunt divizate în două mulțimi. Ca rezultat al împerecherii a doi cromozomi-părinți se obțin doi cromozomi-urmași. În cele mai dese cazuri, primul cromozom-urmaș se obține ca combinație dintre genele din partea stângă a cromozomului-mamă (până la tăietură) și genele din partea dreaptă a cromozomului-tată (de la tăietură); al doilea cromozom-urmaș se obține ca combinație dintre genele din partea stângă (până la tăietură) a cromozomului-tată și genele din partea dreaptă (de la tăietură) a cromozomului-mamă.

Deci, dacă avem:

$$\textit{cromozom-mama } (a_1, a_2, \dots, a_n),$$
$$\textit{cromozom-tata } (b_1, b_2, \dots, b_n),$$

atunci urmașii ce se obțin sunt:

cromozom-urmaș1 $(a_1, a_2, \dots, a_k, b_{k+1}, \dots, b_n)$,

cromozom-urmaș2 $(b_1, b_2, \dots, b_k, a_{k+1}, \dots, a_n)$.

În caz general pot fi aplicate și câteva tăieturi sau pot fi creați urmași ca combinare a mai multor cromozomi.

Asupra cromozomilor-urmași se aplică *mutația* cu o rată ε care costă în alegerea aleatorie a unei gene a cromozomului și modificată aliator valoarea. Acest operator permite depistarea unor cromozomi cărora li se asociază soluții admisibile a problemei formulate care nu pot fi obținute doar prin încrucișare.

Aplicând un algoritm genetic se ține cont de păstrarea unor proprietăți caracteristice care îmbunătățesc rezultatul obținut. Când are loc codificarea problemei, în primul rând se ține cont de faptul că pentru păstrarea unui cromozom să nu se ceară memorie în exces. Un alt aspect este ca timpul de execuție a evaluării funcției obiectiv, încrucișării și mutației să nu fie de ordin înalt. Fiecărui cromozom obținut atât la generarea populației inițiale dar și după aplicarea operatorului de încrucișare sau mutație trebuie să-i corespundă o soluție din domeniul de definiție, astfel se garantează corectitudinea aplicării algoritmului.

Deși este admisă posibilitatea de decodificare *unu la unu* (unui cromozom îi corespunde doar o soluție), *unu la n* (unui cromozom îi corespund *n* soluții) și *n la unu* (la *n* cromozomi le corespunde o soluție) se recomandă aplicarea decodificării *unu la unu*. În algoritmii genetici descriși în lucrare este aplicat modelul unu la unu.

Când vorbim de soluția obținută în urma executării unui algoritm genetic, se ține cont de o balanță între explorarea unui număr cât mai larg de soluții admisibile și exploatarea sa în sensul cât de aproape este soluția obținută de soluția optimă globală. Aici ne referim la alegerea cu atenție a dimensiunii populației generate la pasul inițial care va depinde de dimensiunile rețelei ce descrie problema formulată. Un alt aspect este numărul de iterații după care algoritmul este stopat iar soluția asociată cromozomului cu cele mai bune caracteristici, din ultima populație generată, este acceptată ca soluție optimă a problemei formulate. Un număr mare de iterații presupune că soluția generată se apropie foarte mult de optimul global dar în cazul problemelor de dimensiuni mari numărul mare de iterații presupune și timp mai mare de execuție. Din acest motiv trebuie de găsit un echilibru care ar permite obținerea soluției problemei în timp rezonabil chiar și pentru probleme de dimensiuni mari.

Având la bază cele descrise în literatura de specialitate, se poate concluziona că un algoritm genetic presupune îndeplinirea unor pași care trebuie să se finalizeze cu o soluție optimă (Moanță, 1998), (Eroglu & Adiguzel, 2006):

Pasul 1. Generarea populației inițiale.

Pasul 2. Decodificarea cromozomilor și evaluarea funcției obiectiv pentru fiecare cromozom al populației.

Pasul 3. Selectarea cromozomilor care sunt transferați în următoarea populație, având în vedere valoarea funcției obiectiv, astfel încât să nu fie pierdute soluțiile asociate cromozomilor cu caracteristici bune.

Pasul 4. Încrucișarea cromozomilor selectați pentru obținerea cromozomilor-urmași.

Pasul 5. Mutația unei gene a cromozomului-urmaș cu o rată dată ε.

Pasul 6. Verificarea condiției de oprire. În cazul în care condiția de oprire nu se îndeplinește, se trece la *Pasul 2*. În caz că se îndeplinește, atunci STOP.

Algoritmul genetic este un algoritm evolutiv clasic bazat pe întâmplare, ceea ce presupune că, pentru a găsi o soluție folosind algoritmii genetici soluțiilor actuale li se aplică modificări aleatorii pentru a genera alte soluții noi.

1.5 Problema de transport cu mai mulți indici

În zilele noastre, problema transportului produselor este una actuală, avându-se în vedere globalizarea și faptul că oamenii tind să cumpere produse din diferite zone ale lumii, iar prețul depinde de mai mulți factori care trebuie luați în calcul. În problema de transport se încadrează un număr larg de probleme din viața reală în care ca elemente de bază care o descriu sunt sursele și destinațiile.

Din practică se cunoaște că, în cele mai dese cazuri, problema de transport poate fi descrisă de tipul de produse transportate, de tipul de transport utilizat, de distanța parcursă sau de tipul ambalajului utilizat. Ca urmare a acestor restricții care sunt iminente (inevitabile) la formularea problemei de transport, pentru a nu pierde din corectitudinea formulării problemelor luate din viața reală, se pot diviza problemele de transport în 4 grupuri mari:

1. *Probleme de transport cu doi indici (PT2I),* presupun cunoașterea disponibilităților de produs în surse și cunoașterea necesarului de produs în destinații. Din acest grup face parte problema clasică de transport studiată în secțiunea 1.1 al acestei lucrări. Aici este descris modelul matematic al problemei, sunt nominalizați autorii și lucrările în care aceștia propun metode de soluționare. În secțiunile 1.2 – 1.3 sunt formulate un șir de cazuri particulare ale problemei de bază pentru care autorii propun în lucrările lor diferite metode de soluționare. În special sunt discutate cazurile când funcțiile de cost al cheltuielilor de transport sunt descrise atât de funcții constante sau liniare, cât și de funcții neliniare diferențiabile, ne diferențiabile și separabile.

În capitolul doi al acestei lucrări, se propun algoritmi de soluționare pentru unele cazuri particulare a PT2I, ținând cont de faptul că funcțiile de cost sunt funcții concave. Pentru soluționarea problemei de transport pe rețea cu o sursă și o destinație, a problemei de transport pe rețea cu o sursă și câteva destinații, a problemei de transport pe rețea cu câteva surse și câteva destinații este studiată posibilitatea reducerii problemei neliniare obținute la o problemă liniară. O altă abordare este codificarea fiecărei probleme formulate, astfel încât să fie posibilă aplicarea algoritmilor genetici care generează soluție optimă în timp rezonabil chiar și pentru probleme de transport de dimensiuni mari.

Printre direcțiile de cercetare de care sunt interesați matematicienii în ceea ce privește PT2I sunt: determinarea unei soluții inițiale bune, soluționarea problemei de transport cu constrângeri ale capacității rutei de transportare, cazul când costul transportului unitar, oferta și cererea este fuzzy, dar și soluționarea problemei multiobiectiv.

2. *Probleme de transport cu trei indici (PT3I),* presupun că se cunoaște disponibilitatea de produs în surse, necesarul de produs în destinații și tipurile de produse care trebuie transportate sau tipurile de transport utilizat pentru transportarea produselor. Astfel de probleme pot fi soluționate aplicând o metodă extinsă a potențialilor. Printre lucrările din ultimii ani sunt de menționat (Khurana, Aldakha, & Lev, 2018), (Singh, Chauhen, & Kuldeep, 2018), (Ahmed, Tanvir, Sultana, Mahmud, & Uddin, 2014), (Senapati, 2018). Printre direcțiile de cercetare se poate evidenția problema fuzzy și cea multiobiectiv.

3. *Probleme de transport cu patru indici (PT4I)* presupun că se cunoaște disponibilitatea de produs în surse, necesarul de produs în destinații, tipurile de produse care trebuie transportate și tipurile de transport care pot fi utilizate pentru transportarea produselor de la surse la destinații.

O problemă din acest grup este formulată în 3.1, din capitolul trei al acestei lucrări, pentru care se cunoaște și faptul că funcția de cost este o funcție neliniară - sumă de funcții concave. Se propune reducerea problemei neliniare de transport la o problemă liniară pentru care este aplicată metoda simplex de soluționare. O altă abordare este codificarea problemei astfel încât este posibilă aplicarea unui algoritm genetic de soluționare care generează soluție optimă în timp rezonabil chiar și pentru probleme de dimensiuni mari.

Problemele din acest grup descriu situația reală ce ține de activitatea unei întreprinderi, fapt ce suscită (provoacă) interesul cercetătorilor de a le studia. Este cunoscută situația din economie când un grup de întreprinderi deține materiile prime necesare altui grup de întreprinderi pentru a fabrica/produce produse necesare în viața de zi cu zi. Acest tip de probleme a fost studiat de R. Zitouni, (Zitouni, Keraghel, & Benterki, 2007) care propune un algoritm de soluționare și îl compară cu alte metode în (Zitouni & Achache, 2017). O modificare a algoritmului clasic este

descris de A. Djamel et. al. (2012). T-H. Pham și Ph. Dott (2012), (2013) propun o metodă exactă de soluționare a problemei. O abordare interesantă a PT4I bicriteriale a fost realizată în (Bakhayt, 2016), în care autorul descrie un algoritm genetic de soluționare.

4. *Probleme de transport cu mai mulți indici (multiindex) (PTnI)*. Cazul general al PTnI a fost formulat de Ph.-X. Ninh (1979), care a propus și o metodă exactă de soluționare, o extensie a metodei potențialilor prin coordonarea soluționării problemei primare și a celei duale. Un rezultat important este formularea și demonstrarea teoremei condiției necesare și suficiente de existență a soluției. O lucrare importantă este cea a lui I. M. Stancu-Minasian (1997) care studiază câteva cazuri particulare a problemei de transport dar și un interes deosebit o are lucrarea lui (Раскин Л. Г., 2012). Printre alți autori care au studiat problema de transport cu mai mulți indici se numără K. B. Hayley (1962), care a prezentat un algoritm de rezolvare, W. Junginer (1993), care a efectuat un studiu privind caracteristicile problemei multiindex și a propus un șir de probleme logice pentru soluționarea problemelor de transport cu mai mulți indici.

1.6 Concluzii la capitolul 1

Capitolul 1 conține o analiză succintă a unui șir de probleme de transport care pot fi formulate ca probleme de programare liniară și neliniară descrise de funcții diferențiabile, ne diferențiabile și separabile, dar și a problemelor de transport descrise de mai mulți indici, precum și a algoritmilor propuși de mai mulți autori pentru soluționarea diferitor cazuri particulare. Un șir de algoritmi de determinare a costului minim de transport al unui flux de produs omogen în rețele și de algoritmi genetici care pot fi aplicați pentru soluționarea problemei neliniare de transport sunt studiați pentru a depista posibilități de aplicare a acestor algoritmi în cazul soluționării problemelor nerezolvate sau pentru îmbunătățirea soluțiilor propuse anterior.

Existența *mai multor puncte extreme,* pentru problema studiată în lucrare, poate duce la obținerea unei soluții locale. Deci, aplicarea unor tehnici, cum ar fi condițiile de optimalitate Kuhn-Tucker și multiplicatorii Lagrange, aplicarea derivatelor de ordinul întâi (metode gradient) și a celor de ordinul doi (matricea Hessian) dar și a funcțiilor de penalitate devin greoaie sau sunt deseori chiar impracticabile.

În baza analizei situației actuale în domeniul problemelor neliniare de transport se formulează următoarele concluzii:

1. Problema neliniară de transport este o problemă actuală importantă care este studiată de un număr impunător de matematicieni, astfel încât să fie posibilă implementarea soluțiilor propuse în cazul problemelor reale.

2. Deoarece problema neliniară de transport cu funcții concave de cost este NP-dificilă, este importantă dezvoltarea algoritmilor care ar furniza soluții în timp rezonabil.
3. Un aspect important este faptul că funcțiile obiectiv ale problemelor neliniare de transport sunt funcții separabile care sugerează posibilitatea reducerii problemelor neliniare complexe la probleme de programare liniară.
4. O atenție deosebită merită algoritmii genetici care pot fi utilizați la soluționarea problemelor neliniare de transport, pentru a evita utilizarea derivatelor și a Hessianului. Deși astfel de algoritmi nu pot garanta obținerea unei soluții optime globale, ei sunt potriviți pentru cazul funcțiilor obiectiv complexe și/sau pentru cazul problemelor de dimensiuni mari, generând soluții care sunt în vecinătate apropiată a soluțiilor globale în timp rezonabil.
5. Deoarece un element important al algoritmilor genetici este codificarea corectă a cromozomilor care descriu populațiile astfel încât la decodificare se obțin mulțimi de soluții admisibile, este recomandabil de a utiliza așa elemente ale teoriei grafurilor, ca: arbore de acoperire, drum între două puncte, existența unui flux în rețea, căutarea în adâncime.
6. Un interes deosebit prezintă problemele neliniare de transport cu funții concave de cost cu câteva tipuri de produse și cu câteva tipuri de transport, care pot fi formulate ca probleme cu mai mulți indici și care necesită descrierea unor algoritmi ce permit și soluționarea problemelor de dimensiuni mari.
7. Problema de transport pe rețea cu funcții concave de cost cu mai mulți indici necesită a fi descrisă matematic, urmând a fi propuși algoritmi de soluționare care ar putea fi aplicați și pentru probleme de dimensiuni mari.

II. PROBLEMA NELINIARĂ DE TRANSPORT CU FUNCȚII CONCAVE DE COST

În acest capitol sunt prezentate rezultatele de bază ce se referă la algoritmii de soluționare a problemei de transport pe rețea cu funcții concave de cost, care presupune minimizarea costului total de transport a unui flux omogen de produs din sursă/surse în destinație/destinații. Sunt examinate câteva cazuri particulare ale problemei: *cazul rețelei de transport cu o sursă și o destinație, cazul rețelei de transport cu o sursă și o destinație în care se impune restricția că fluxul de produs este transportat prin toate arcele rețelei, cazul rețelei de transport cu o sursă și câteva destinații și cazul rețelei d transport cu câteva surse și câteva destinații*.

Problema este studiată din perspectiva obținerii unor algoritmi eficienți, nu neapărat exacți, care ar permite soluționarea în timp rezonabil al problemelor de dimensiuni mari, indiferent de faptul că problema studiată este una NP-dificilă.

Pentru o astfel de problemă de minimizare a costului de transport și pentru o mare parte dintre cazurile ei particulare, în care funcția obiectiv este una neliniară concavă, algoritmii exacți cer un timp exponențial de execuție și adeseori ei obțin în calitate de soluție unul dintre minimele locale, care nu neapărat este și minim global.

În continuare, sunt expuși câțiva algoritmi care generează soluții pseudo optimale în timp rezonabil. Algoritmii sunt implementați în Sistemul Mathematica, fiind testați pe rețele de transport de diferite dimensiuni. Rezultatele sunt publicate într-o serie de lucrări (Lozovanu & Pasha, 2002), (Pașa & Ungureanu, 2017a), (Pașa & Ungureanu, 2017d), (Pașa & Ungureanu, 2017e), (Pașa & Ungureanu, 2017f), (Pașa, 2018c), (Pașa, 2018d), (Pașa, 2020a), (Pașa, 2020c).

2.1 Formularea problemei. Proprietăți

Se consideră problema neliniară de transport pe rețea descrisă de graful conex aciclic $G = (V, E)$, $|V| = n$, $|E| = m$, unde pe mulțimea de vârfuri este definită funcția de producere și consum $q: V \to R$, iar pe mulțimea de arce sunt definite funcțiile neliniare concave nedescrescătoare de cost $\varphi_e(x_e)$. Problema neliniară de transport pe rețea constă în determinarea unui flux x^* care minimizează funcția $F(x) = \sum_{e \in E} \varphi_e(x_e)$ care descrie costul de transport, adică se cere soluționarea problemei:

$$F(x) \to min, \tag{2.1}$$

$$\sum_{e \in E^+(v)} x(e) - \sum_{e \in E^-(v)} x(e) = q(v), \tag{2.2}$$

$$x(e) \geq 0, \quad e \in E, \tag{2.3}$$

unde $E^-(v) = \{(v,u) | (v,u) \in E\}$, $E^+(v) = \{(u,v) | (u,v) \in E\}$, iar sistemul de restricții definește mulțimea de soluții admisibile X ale problemei formulate, adică soluțiile care satisfac sistemul de ecuații și condițiile de nenegativitate (Lozovanu & Pasha, 2002).

Se presupune că toate datele sunt valori reale și se dorește să se obțină ca soluție optimă o valoare reală fezabilă.

Problema de transport pe rețea este formulată în așa fel încât pe oricare dintre arcele rețelei să poată fi transportat orice volum de produs, costul de transport fiind descris de o funcție concavă în raport cu volumul produsului transportat pe arc. În plus, se consideră că volumul de produs care se cere a fi transportat nu ajunge direct de la surse la destinații, ci trece prin puncte intermediare în care volumul de produs nu se mărește și nici nu se micșorează (nu se produce nimic și nici nu se consumă nimic) – e satisfăcută condiția de conservare a fluxului de produs care trece prin rețea. Se presupune că rețeaua de transport definită pe graful G admite fluxul specificat. Existența fluxului în rețea este studiată în (Ford & Fulkerson, 1962).

Fie x un flux în G. Se notează prin $G_x = (V_x, E_x)$ subgraful generat de arcele $e \in E$ pentru care $x(e) > 0$. Fie $X(G,q) \subseteq \mathbb{R}^m$ - mulțimea soluțiilor admisibile pentru problema neliniară de transport descrisă de graful G și funcția de producere și consum q.

Se scot în evidență câteva proprietăți (Лозовану, 1991) ale problemei formulate.

Lema 2.1. *Mulțimea $X(G,q) \subseteq \mathbb{R}^m$ este mărginită atunci și numai atunci când G nu conține cicluri.*

Lema 2.2. *Dacă $x' = (x'(e_1), x'(e_2), \dots, x'(e_m))$ este punct de extrem al mulțimii convexe poliedrale $X(G,q) \subseteq \mathbb{R}^m$, atunci graful respectiv $G_{x'} = (V_{x'}, E_{x'})$ nu conține cicluri.*

Teorema 2.1. *Dacă G nu conține cicluri, atunci pentru funcțiile concave φ_e, $e \in E$, în mulțimea de fluxuri optimale ale problemei de transport pe rețea există fluxul x^* pentru care G_{x^*} nu conține cicluri.*

Pentru determinarea aciclicității unui graf poate fi utilizată tehnica căutării în adâncime (depth first search − DFS). Dacă pe parcursul traversării grafului G prin DFS se întâlnește cel puțin un arc opus, graful conține cel puțin un ciclu. Reciproca de asemenea este adevărată: dacă un graf orientat conține cel puțin un ciclu, atunci în orice traversare a grafului prin DFS apare cel puțin un arc opus.

Teorema 2.2. *Problema de transport pe rețea cu funcții concave de cost* φ_e, $e \in E$ *este o problemă NP-completă.*

În concluzie se poate spune că problema de transport pe rețea cu funcții concave de cost poate fi soluționată utilizând algoritmi finiți care se bazează pe cercetarea tuturor grafurilor fără cicluri, ce descriu soluții admisibile. Fiecărui graf îi este asociat un flux pentru care se calculează costul de transport descris de valoarea funcției obiectiv.

Soluționând sistemul de ecuații (2.2), se obține o soluție generală în care n necunoscute bazice se exprimă liniar prin celelalte $m - n$ necunoscute secundare. O soluție particulară care se obține când se atribuie valoarea zero tuturor variabilelor secundare în expresia pentru soluția generală se numește *soluție de bază*. Deoarece se pot forma grupuri de necunoscute bazice ale sistemului a câte n necunoscute, numărul soluțiilor de bază este finit și nu depășește C_m^n. Modificând pe rând necunoscutele bazice, se pot obținute toate soluțiile de bază ale sistemului (2.2). Dacă toate componentele soluției particulare sunt numere nenegative ea se numește *soluție admisibilă*. Mulțimea soluțiilor admisibile formează poliedrul (mulțimea poliedrală a) soluțiilor admisibile ale sistemului (2.2) – (2.3). Între vârfurile poliedrului soluțiilor admisibile și soluțiile admisibile de bază ale sistemului liniar considerat există o corespondență biunivocă. Din acest motiv, numărul de vârfuri ale poliedrului soluțiilor admisibile coincide cu numărul soluțiilor admisibile de bază ale sistemului.

Vectorii $x = (x(e_1), x(e_2), \dots, x(e_m))$ care satisfac condiția (2.2) – (2.3) (Гольштейн & Юдин, 1969) formează în spațiul m-dimensional o mulțime convexă poliedrală $X(G, q)$. Deoarece se presupune că rețeaua G admite flux, mulțimea $X(G, q) \neq \emptyset$.

Teorema 2.3. *Mulțimea soluțiilor admisibile ale unui sistem de ecuații și inecuații liniare reprezintă o mulțime poliedrală convexă.*

Teorema 2.4. *Există o corespondență biunivocă între soluțiile admisibile de bază ale unui sistem de ecuații și inecuații liniare și vârfurile mulțimii poliedrale de soluții admisibile ale aceluiași sistem. Fiecare vârf reprezentă o soluție admisibilă de bază a sistemului, și reciproc.*

Punctul de minim global al unei funcții este printre punctele de minim local. Este clar că dacă o soluție $x = (x_1, x_2, .., x_m)$ este punct care satisface restricțiile de minim global, atunci $x = (x_1, x_2, .., x_m)$ este și minim local.

2.2 Soluționarea problemei neliniare de transport

O mare parte dintre metodele de soluționare a problemei de programare neliniară găsesc soluția optimă globală ca rezultat al determinării iterative a unui șir de soluții admisibile. Utilizatorul indică când procesul de generare a șirului urmează să fie oprit și care dintre soluțiile admisibile trebuie să fie considerată optimală. Dacă nu este garantată optimalitatea globală, atunci soluția găsită este doar una pseudo optimală. Adesea, în cazul unor probleme de asemenea tip, criteriul de oprire a procesului de calcul este bazat nu pe considerente teoretice, ci pe considerente ce țin de timpul de calcul, de memoria internă și externă disponibilă, de complexitatea calculelor efectuate, de alți factori.

2.2.1 Algoritmii AE1 și AE2

Un aspect caracteristic al problemelor de optimizare neliniară este faptul că ele pot avea *mai multe optime locale*. Dacă problema nu este una de optimizare convexă, atunci aflarea optimului global ar putea fi efectuată și prin trierea soluțiilor optime locale. Dacă timpul de calcul este unul considerabil, în calitate de soluție acceptată pentru problema soluționată poate fi luată cea mai bună dintre soluțiile locale determinate până la momentul de oprire a procesului de calcul, deci cea căreia îi corespunde valoarea minimă a funcției obiectiv. Evident că în cazul aplicării unui asemenea criteriu de oprire a procesului de calcul, nu se garantează că soluția găsită este una globală, adică este mai degrabă o soluție pseudo optimală.

Situația descrisă este comună și pentru procesul de soluționare a problemei neliniare (2.1)-(2.3), care este o problemă de transport pe rețea ce are drept scop determinarea costului minim de transport a fluxului de produs, utilizând algoritmul propus în (Lozovanu & Pasha, 2002) și îmbunătățit ulterior în (Pașa & Ungureanu, 2017e).

Algoritmul constă în reducerea consecutivă a problemei de optimizare liniară la o problemă a programării liniare. Pentru determinarea coeficienților care descriu funcția obiectiv liniară la care este redusă funcția obiectiv neliniară a problemei inițiale se folosește schema de soluționare a problemelor separabile deoarece ea satisface condițiile.

Algoritmul AE1

Pasul 1. Se construiește o soluție admisibilă inițială $x^0 = \left(x^0(e_1), x^0(e_2), \dots, x^0(e_m)\right)$ a sistemului:

$$\begin{cases} \sum\limits_{e \in E^+(v)} x(e) - \sum\limits_{e \in E^-(v)} x(e) = q(v), & v \in V, \\ x(e) \geq 0, & e \in E. \end{cases} \tag{2.4}$$

Pasul 2. Se determină valoarea funcției obiectiv în x^0:

$$F(x^0) = \sum_{e \in E} \varphi_e\left(x^0(e)\right), \tag{2.5}$$

și se calculează valoarea coeficienților:

$$C_e = \begin{cases} \dfrac{\varphi_e\left(x^0(e)\right)}{x^0(e)}, & x^0(e) > 0, \\ \varphi_e'(0), & x^0(e) = 0, \end{cases} \tag{2.6}$$

pentru fiecare $e \in E$.

Pasul 3. Se soluționează problema liniară de transport:

$$z(x) = \sum_{e \in E} C_e x(e) \to min,$$
$$\begin{cases} \sum\limits_{e \in E^+(v)} x(e) - \sum\limits_{e \in E^-(v)} x(e) = q(v), & v \in V, \\ x(e) \geq 0, & e \in E, \end{cases} \tag{2.7}$$

și se obține soluția optimă $x^1 = \left(x^1(e_1), x^1(e_2), \dots, x^1(e_m)\right)$ a problemei liniare.

Pasul 4. Se compară valorile $z(x^1)$ și $F(x^0)$. Dacă $z(x^1) < F(x^0)$ sau $z(x^1) = F(x^0)$ și $x^1 \neq x^0$, atunci x^0 se substituie cu x^1 și se trece la *Pasul 2*, ceia ce înseamnă că se repornește procedura de liniarizare cu o altă soluție inițială. Dacă $z(x^1) > F(x^0)$ sau $z(x^1) = F(x^0)$ și $x^1 = x^0$, atunci în calitate de soluție optimă a problemei neliniare este considerată soluția $x^* = x^0$. **STOP.**

În Anexa 3 este prezentată schema-bloc a algoritmului AE1 descris mai sus.

În (Pașa & Ungureanu, 2017a) este formulat și descris algoritmul de soluționare a problemei de transport pe rețea pentru cazul când fluxul pe arcele rețelei este restricționat de capacitățile lor, care indică limitele de jos și de sus ale fluxurilor pe arce, iar funcțiile de cost sunt concave secvențial-liniare.

În urma testelor efectuate pe mai multe exemple de rețele de transport de diferite dimensiuni, s-a constatat că soluția obținută depinde de soluția inițială de la *Pasul 1*, adică de soluția sistemului (2.4). Soluția inițială poate fi determinată diferit în dependență de metoda utilizată pentru soluționarea sistemului (2.4).

În acest context apare ideea ca algoritmul de soluționare a problemei de transport pe rețea cu funcții concave de cost să fie modificat astfel, încât să pornească de la mai multe soluții admisibile, iar dintre soluțiile optime, obținute în urma executării algoritmului, să fie aleasă acea

care corespunde valorii minime a funcției obiectiv. De remarcat faptul că este posibil de a efectua calcule în paralel. Pentru un asemenea algoritm modificat se obține o soluție pseudo optimă cu caracteristici mai bune, indiferent de structura și complexitatea rețelei, dimensiunile grafului care descrie rețeaua de transport sau de funcțiile concave de cost asociate arcelor.

În baza celor expuse, algoritmul inițial este îmbunătățit prin faptul că pot fi efectuați în paralel pașii de determinare a soluției optime pentru cazul a câtorva soluții inițiale, ca apoi să se selecteze cea mai bună soluție dintre cele determinate, care și va fi soluția pseudo optimă a problemei formulate.

Algoritmul AE2 *cu câteva soluții inițiale*

Faza I Inițializarea datelor.

Pasul 1. Se construiește o mulțime ce conține p soluții admisibile inițiale de forma $x^0 = \left(x^0(e_1), x^0(e_2), \dots, x^0(e_m)\right)$, soluții ale sistemului (2.4). Faza II se execută pentru fiecare dintre cele p soluții admisibile inițiale obținute.

Faza II Se execută pentru fiecare element al mulțimii.

Pasul 2. Se determină valoarea funcției obiectiv în x^0 (2.5) și se calculează valoarea coeficienților (2.6) pentru fiecare $e \in E$.

Pasul 3. Se soluționează problema liniară de transport (2.7) și se obține soluția optimă $x^1 = \left(x^1(e_1), x^1(e_2), \dots, x^1(e_m)\right)$.

Pasul 4. Se compară valorile $z(x^1)$ și $F(x^0)$. Dacă $z(x^1) < F(x^0)$ sau $z(x^1) = F(x^0)$ și $x^1 \neq x^0$, atunci x^0 se substituie cu x^1 și se trece la *Pasul 2* ceea ce înseamnă că se repornește procedura de liniarizare cu altă soluție inițială. Dacă $z(x^1) > F(x^0)$ sau $z(x^1) = F(x^0)$ și $x^1 = x^0$, atunci soluția optimă a problemei neliniare este considerată valoarea $x^* = x^0$.

Faza III Se obține soluția problemei.

Pasul 5. Se compară valorile funcției obiectiv în toate soluțiile obținute și se determină valoarea sa minimală. În calitate de soluție a problemei servește cea căreia îi corespunde valoarea minimă a funcției obiectiv. **STOP**.

Remarca 2.1. *În cazul în care sunt impuse restricții asupra volumului de produs transportat pe arcele rețelei, acestea pot fi adăugate la sistemul de restricții. Sistemul (2.4) este înlocuit cu sistemul:*

$$\begin{cases} \sum_{e \in E^+(v)} x(e) - \sum_{e \in E^-(v)} x(e) = q(v), & v \in V, \\ l(e) \le x(e) \le u(e), & e \in E, \end{cases} \quad (2.4^*)$$

unde $l(e)$ este limita de jos, iar $u(e)$ este limita de sus a volumului de flux ce poate fi transportat pe arcul $e \in E$.

Remarca 2.2. *Algoritmii AE1 și AE2 pot fi aplicați în cazul rețelelor de transport cu una sau mai multe surse și cu una sau mai multe destinații.*

Se formulează și se demonstrează următoarele teoreme:

Teorema 2.5. *Algoritmul AE1 converge către un optim local.*

Demonstrație. *Algoritmul AE1 pornește de la o soluție inițială admisibilă care este unul dintre vârfurile poliedrului ce descrie mulțimea soluțiilor admisibile care satisfac sistemul de restricții liniare ale problemei neliniare cu funcții concave de cost. Soluția problemei liniare, care este problema relaxată, satisface aceleași restricții deci face parte din același domeniu de definiție și este asociată cu unul dintre vârfurile poliedrului. Prin urmare, se poate spune că algoritmul converge întotdeauna către un optim local din domeniul de definiție a problemei inițiale neliniare.*□

Teorema 2.6. *Algoritmul AE2 converge către un optim local.*

Demonstrație. *Potrivit Teoremei 2.5 algoritmul AE1 converge către un optim local. Deoarece algoritmul AE2 presupune executarea algoritmului AE1 de câteva ori fiind selectată soluția care corespunde valorii celei mai mici a funcției obiectiv, rezultă că și algoritmul AE2 converge către un optim local.*□

2.2.2 Implementarea și testarea algoritmilor AE1 și AE2

Sistemul Mathematica (Wolfram, 2016) permite aplicarea algoritmului utilizând limbajul Wolfram: cod compact, ușor de citit, simplu pot fi definite variabilele și funcțiile, există o mulțime de funcții standard.

Pentru obținerea soluției inițiale a algoritmului este utilizată funcția standard *FindInstace[]* care soluționează sistemul de ecuații și, ca rezultat, livrează o soluție cu elemente pozitive din mulțimea de soluții ale sistemului. *LiniarPrograming[]* este o funcție standard care permite soluționarea problemei de programare liniară aplicând algoritmul simplex. Funcția standard *AppendTo[]* permite adăugarea unui element nou în mulțimea de soluții. Funcția standard *DeleteDuplicates[]* permite eliminarea din listă a elementelor care se repetă.

Aplicarea algoritmului AE1 pentru soluționarea unei probleme neliniare de transport pe rețea este prezentată în Anexa 4.

Algoritmii propuși sunt comparați cu rezultatele furnizate de funcțiile standard ale Sistemului Mathematica:

- funcția *Minimize[]* – se utilizează pentru obținerea soluției optime globale a problemei de optimizare la care folosesc metodele programării liniare, multiplicatorii Lagrange, metodele programării în numere întregi și alte metode analitice care permit obținerea soluțiilor precise sau, altfel spus, soluții simbolice;
- funcția *NMinimize[]* – se utilizează pentru obținerea soluției optime globale ca o valoare numerică fiind aplicate metodele programării liniare, metoda Nelder-Mead, căutarea aleatorie și alte metode numerice care, în urma unei serii de aproximări, conduc la soluția căutată. Se poate utiliza atributul *Method* pentru alegerea explicită a metodei care soluționează problema, printre care: *DifferentialEvolution*, *RandomSearch*, *SimulatedAnnealing*, *Neldermead*.

Rezultatele obținute aplicând algoritmul AE1 și cele obținute aplicând funcțiile standard ale Sistemului Mathematica sunt prezentate în Tabelul 2.1 pentru probleme descrise de rețele de diferite dimensiuni cu n - vârfuri și m - arce.

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

Tabelul 2.1. Timpul de execuție pentru algoritmul AE1 și funcțiile standard. Sursa: elaborat de autor

n	**m**	**Algoritmul AE1** (secunde)	**Minimize** (secunde)	**NMinimize** (secunde)	**FindMinim** (secunde)
4	***6***	0.000279	0.17	0.2	0.18
6	***10***	0.000360	3.02	0.38	0.12
7	***14***	0.000492	108.45	0.61	0.19
8	***16***	0.000468	345.61	2.09	0.19
8	***20***	0.001135	10386.50	1.96	1.75

Rezultatele prezentate în Tabelul 2.2 sunt obținute generând câte 100 de soluții admisibile inițiale pentru fiecare problemă, soluția optimă fiind selectată printre valorile obținute în rezultatul aplicării algoritmului AE1. Aceeași problemă este soluționată și pentru m soluții admisibile inițiale, unde m este numărul de arce a grafului care descrie rețeaua de transport. Se poate vedea

care este timpul de execuție a algoritmului pentru fiecare caz dar și numărul de soluții pentru care valoarea funcției obiectiv minimă coincide.

Tabelul 2.2. Timpul de execuție pentru algoritmul AE2. Sursa: elaborat de autor

m	Pentru *m* soluții inițiale		Pentru 100 de soluții inițiale		*Minimize[]*
	Timp de execuție (secunde)	**Nr. de soluții optime**	**Timp de execuție** (secunde)	**Nr. de soluții optime**	**Timp de execuție** (secunde)
6	0.0781	1	0.6718	1	0.1280
7	0.9375	3	0.8282	3	0.2332
8	0.2187	1	1.9055	1	0.5189
9	0.4531	1	3.7220	1	1.1229
10	1.1562	2	9.9054	2	2.7257
11	1.9375	2	15.1700	2	17.0920
12	4.4531	4	24.1243	4	32.3139
13	10.0625	2	68.0372	2	72.2206
14	16.4063	4	99.1259	4	68.9724
15	40.6250	2	236.5117	2	81.7750
16	56.1250	2	313.3157	6	345.6110
17	140.9530	2	727.7770	2	308.3290
18	333.2340	2	1599.7834	2	1827.4400

Deși m este o mărime mult mai mică decât 100, în majoritatea cazurilor nu se obțin mai multe soluții pentru problemele formulate, deci se poate spune că pentru obținerea rezultatului așteptat este suficientă utilizarea a m soluții inițiale.

Tradițional, programarea secvențială este aplicată pentru calcule descrise de un șir secvențial de instrucțiuni care sunt executate una după alta de un singur procesor.

Programarea paralelă presupune utilizarea mai multor componente hardware ce permit executarea simultană a unor secvențe divizate în părți discrete, care sunt soluționate în același moment de timp. Fiecare secvență constă dintr-o serie de instrucțiuni, fiecare dintre care se execută simultan pe nuclee sau procesoare diferite. Obiectivul primar al programării paralele este creșterea puterii de calcul pentru o execuție mai rapidă a algoritmilor.

Conform (Flynn, 1972) sistemele computaționale pot fi clasificate în dependență de fluxul de instrucțiuni executate și de fluxul de date ce trebuie procesate astfel:

- SISD (Single Instruction, Single Data) – instrucțiune unică, date unice;

- SIMD (Single Instruction, Multiple Data) – instrucțiune unică, date multiple;
- MISD (Multiple Instruction, Single Data) – instrucțiune multiplă, date unice;
- MIMD (Multiple Instruction, Multiple Data) – instrucțiune multiplă, date multiple.

În contextul problemei formulate este recomandabil de a utiliza sistemele SISD și SIMD în dependență de dimensiunea rețelei de transport, de timpul disponibil pentru obținerea soluției optimale, precum și de posibilitățile tehnice de care se dispune.

Sistemul de calcul SISD este aplicat în implementarea algoritmilor secvențiali, în cazul dat ne referim la algoritmul AE2, deoarece el satisface condiția că o unitate de procesare execută o secvență de instrucțiuni – determinarea soluției optime, și operează cu o singură dată de intrare – cu soluția inițială cu care începe algoritmul pentru determinarea soluției optime.

Deoarece aceeași secvență de instrucțiuni se execută pentru fiecare soluție inițială obținută la *Pasul 1*, se poate spune că pentru algoritmul AE2 este corectă și aplicarea sistemului de calcul SIMD, care utilizează o unitate de control ce dirijează cu N procesoare astfel încât fiecare execută secvența de instrucțiuni descrisă de *Pașii 2 – 4* ai algoritmului. În același timp, drept date de intrare pentru fiecare procesor este o soluție inițială diferită cu care operează. Schimbul de date între procesoare este realizat de memoria comună unde se stochează rezultatele și se utilizează mai târziu pentru executarea *Pasului 5*. Dacă se decide aplicarea programării paralele, trebuie de avut în vedere faptul că îmbunătățirea timpului de execuție este limitat de sistem.

Sistemul Mathematica (Wolfram, 2016) oferă un mediu integrat și automatizat pentru calculul paralel, fără configurări, interactivitate completă, funcționare locală și de rețea fără întreruperi. Caracterul simbolic al limbajului Wolfram oferă posibilitatea utilizării imediate a unei varietăți extinse de paradigme de programare paralelă și a noi modele de partajare a datelor. Printre funcțiile standard se numără *$KernelCount,* care returnează numărul de nuclee active pentru programarea paralelă. Pentru sincronizarea nucleelor poate fi utilizată funcția *WaitAll[{*pid_1*,* pid_2*, …,* pid_n*}]*. Funcția *Parallelize[expr]* evaluează *expr* utilizând paralelizarea automată, iar *ParallelEvaluate[expr]* evaluează expresia *expr* pentru toate nucleele paralele active și returnează lista de rezultate obținute. Funcția care primește ca date de intrare utilizate de nuclee pentru executarea expresiilor este *ParallelEvaluate[expr,{*ker_1*,*ker_2*,… }]*. Funcția *ParallelEvaluate[expr, kernel]* permite să se specifice care nucleu evaluează expresia, iar *ParallelSubmit[expr]* transmite *expr* pentru a fi evaluat următorul nucleu paralel și returnează expresia evaluată.

Aplicarea funcțiilor standard permite îmbunătățirea timpului de execuție a algoritmului, dar aceasta poate fi mai vizibilă la rezolvarea problemelor de dimensiuni mari.

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

Tabelul 2.3. Timpul de execuție pentru algoritmul AE2. Sursa: elaborat de autor

m	Programare secvențială (secunde)		Programare paralelă (secunde)	
	Faza 1,2 și 3	**Faza 2 și 3**	**Faza 1,2 și 3**	**Faza 2 și 3**
8	2.7465	0.0559	2.7415	0.1199
10	8.2757	0.0509	8.1807	0.1359
12	45.3851	0.0509	45.0851	0.1329
14	141.017	0.0529	139.779	0.1539
16	428.756	0.0489	427.229	0.1989
18	2920.07	0.0409	2944.071	0.1999

Din Tabelul 2.3 se observă că cea mai mare parte de timp este necesară pentru generarea soluției inițiale. Dacă se compară timpul de execuție a două faze (fazele II și III) ale algoritmului AE2 executat secvențial și paralel se observă că el este de același ordin.

2.3 Soluționarea problemei neliniare de transport pe rețea standard utilizând algoritmi genetici

Soluționarea problemei de optimizare neliniară pe rețele de transport de dimensiuni mari prin metoda trierii tuturor soluțiilor admisibile solicită un timp exponențial și echipament de calcul extrem de performant. Pentru asemenea probleme pot fi aplicați algoritmii genetici care permit obținerea unor rezultate acceptabile în timp rezonabil.

Se preferă utilizarea unui model elitist a algoritmului genetic, ceea ce implică păstrarea cromozomilor promițători dintr-o populație pentru populația următoare. Astfel, nu sunt pierdute soluțiile ce corespund cromozomilor promițători ca rezultat al operatorilor de selecție, încrucișare sau mutație. Cromozomii pierduți prin astfel de operații, în cele mai dese cazuri, nu pot fi restabiliți mai târziu.

Selecția cromozomilor presupune sortarea lor în ordinea crescătoare a valorii funcției obiectiv calculată în soluția asociată cromozomilor respectivi. Cromozomii din prima jumătate a populației sunt transferați în populația nouă. Astfel se evită posibilitatea pierderii unei soluții asociate unui cromozom cu caracteristici bune, care ar putea apărea chiar în una dintre primele populații. Acești cromozomi participă în calitate de părinți pentru cromozomii-urmași ai populației noi. Deși astfel de selecție nu garantează obținerea soluției optime globale, este totuși o îmbunătățire față de selectarea aleatorie a cromozomilor-părinți.

A doua jumătate a populației noi o formează cromozomii-urmași obținuți prin *încrucișarea* cromozomilor selectați, iar în calitate de cromozomi-părinți se iau la rând câte doi cromozomi

cărora li se aplică o tăietură aleatorie. Primul cromozom-urmaș conține genele din partea stângă a cromozomului-mamă (până la tăietură) și genele din partea dreaptă a cromozomului-tată (de la tăietură), iar al doilea cromozom-urmaș conține genele din partea stângă a cromozomului-tată și genele din partea dreaptă a cromozomului-mamă. În rezultatul încrucișării a doi cromozomi-părinți se obțin doi cromozomi-urmași. Cromozomii-urmași obținuți sunt, în cele mai dese cazuri, la fel de promițători ca și cromozomii-părinți.

Modificarea aleatorie a unei gene a cromozomului, numită în cadrul descrierii algoritmului genetic *mutație*, poate îmbunătăți soluția dar poate și s-o înrăutățească. Mutația are loc cu o probabilitate ε și nu permite blocarea algoritmului într-o soluție locală. Mutația permite introducerea unor soluții noi, în mulțimea de soluții admisibile studiate, care altfel nu pot fi obținute. În cadrul algoritmului, mutația constă în generarea unei valori noi pentru o genă aleasă aleatoriu din cromozomul-urmaș.

În calitate de condiție de oprire poate servi cerința ca execuția algoritmului să dureze până când sunt generate k populații, deci executate k iterații. În calitate de soluție a problemei de neliniare de transport se acceptă soluția asociată cromozomului cu cele mai bune caracteristici din ultima populație, soluție pentru care valoarea funcției obiectiv este minimă. E posibilă situația în care, începând cu o populație generată la unul dintre pași, următoarele populații conțin aceiași cromozomi și nu există posibilitatea de apariție a unor noi cromozomi. O asemenea situație e caracteristică în special problemelor de dimensiuni mici. Pentru evitarea sau depășirea ei, se impune respectarea unei condiții de tipul $|f(i-1) - f(i)| \leq \varepsilon$, pentru soluțiile asociate cromozomilor cu cele mai bune caracteristici din două populații consecutive $P(i-1)$ și $P(i)$.

2.3.1 Algoritmul genetic AG1

În cele ce urmează este studiată problema de optimizare neliniară de transport pe rețea de tipul (2.1) – (2.3) cu funcția de producere și consum de forma:

$$q(v) = 0, \text{ pentru } v \in V \setminus \{v_0, v_t\} \text{ și } q(v_0) = -q(v_t) \tag{2.8}$$

unde v_0 este sursa rețelei, $v_t -$ destinația ei.

Astfel, se cere determinarea costului minim de transport în rețea cu o sursă și o destinație pentru cazul când costul este descris de o funcție concavă nedescrescătoare.

Deoarece populația generată trebuie să conțină un număr reprezentativ de cromozomi, care ar permite se fie generată soluția problemei în timp rezonabil, se consideră o populație cu $4n$ cromozomi. Multiplul lui 4 permite să se selecteze o jumătate din populație pentru a o transfera în populația nouă fără a avea probleme de divizibilitate la 2. Pe de altă parte, din cei $2n$ cromozomi

transferați în populația nouă, care participă la încrucișare, ușor se pot forma n perechi de cromozomi-părinți. Ei generează alți $2n$ cromozomi-urmași ca urmare a aplicării încrucișării pentru a completa populația nouă.

Populația evoluează către populații ce conțin cromozomi cu caracteristici mai bune, fiind aplicate operațiile de selecție, încrucișare și mutație.

Algoritmul genetic de soluționare a problemei neliniare de transport începe cu o populație generată aleatoriu, fiecare cromozom al căreia reprezintă o codificare a unei soluții admisibile a problemei. Fiecare cromozom este de dimensiunea n, deci conține n elemente numite în continuare - gene. Algoritmul generează la fiecare iterație cromozomi cu caracteristici mai bune decât ale celor din populația precedentă, deoarece trecând de la o populație la alta sunt selectați cromozomii promițători din populația precedentă, noua populație fiind completată cu cromozomi-părinți și cu cromozomi-urmași. Populația este păstrată ca o matrice bidimensională de dimensiunea $4n \times n$ în care fiecare linie este un cromozom.

Pentru simplitate, în descrierea algoritmului, se presupune că primul vârf din rețea v_1 este sursa rețelei, iar ultimul vârf v_n – destinația.

Algoritmul genetic AG1 propus pentru soluționarea problemei de transport cu funcții concave nedescrescătoare secvențial-liniare constă din următorii pași:

Pasul 1 *Inițializarea.* Se generează populația inițială din $4n$ cromozomi. Fiecare cromozom este descris de o mulțime $Nr = \{nr_1, nr_2, \dots, nr_n\}$ care conține n numere întregi și fiecare element reprezintă numărul de arce care ies din vârful respectiv. Astfel:

- pentru fiecare poziție $i = \overline{1, n-1}$ este generat un număr întreg pozitiv nr_i între 1 și numărul de arce ce ies din vârful i.
- pe ultima poziție este plasat numărul $nr_n = 0$ (din *destinație* nu iese niciun arc).

Cele nr_i, $i = \overline{1, n-1}$ arce care ies din vârful i, pe care ulterior este transportat flux de produs, sunt selectate aleatoriu.

Partea de flux care trece prin arcul (i, j), din totalul fluxului care intră în vârful i, este dat de matricea proporțiilor de repartiție $R = \{r_{ij} | i = \overline{1, n}, j = \overline{1, n}\}$ generată aleatoriu, și pentru care se respectă condiția $\sum_{j=1}^{n} r_{ij} = 1, i = \overline{1, n-1}$, și $r_{ij} = 0$, pentru $i = n, j = \overline{1, n}$.

Pasul 2 *Decodificarea și Evaluarea cromozomilor.* Decodificarea presupune asocierea fiecărui cromozom cu o soluție admisibilă de forma $x = (x_1, x_2, \dots, x_m)$, unde x_h, $h = \overline{1, m}$ este cantitatea de flux care trece prin arcul (i, j) calculată astfel: $x_h = r_{ij} * cf_i$, unde cf_i este cantitatea de flux care intră în vârful i și $x_h = 0$ dacă prin arcul respectiv nu trece flux.

Evaluarea presupune determinarea valorii funcției obiectiv pentru fiecare dintre soluțiile obținute.

Pasul 3 *Selectarea.* Cromozomii sunt sortați în ordinea creșterii valorii funcției obiectiv în soluția asociată cromozomului. Cromozomii din prima jumătate a populației, adică $2n$ cromozomi, sunt transferați în noua populație.

Pasul 4 *Încrucișarea cromozomilor.* Încrucișarea are loc între cromozomii transferați în populația $P(i)$ din populația $P(i-1)$. Aleatoriu este aplicată aceeași tăietură ambilor cromozomi-părinți luați câte doi. După tăietură, se obțin mulțimile Nr_1 și Nr_2, $Nr = Nr_1 \cup Nr_2$, $Nr_1 = \{nr_1, nr_2, \dots, nr_k\}$, $Nr_2 = \{nr_{k+1}, nr_{k+2}, \dots, nr_n\}$. Nr_1 – partea stângă iar Nr_2 - partea dreaptă a cromozomului.

Fiecare pereche de cromozomi-părinți generează doi cromozomi-urmași astfel:

- primul cromozom-urmaș primește partea stângă a cromozomului-mamă și partea dreaptă a cromozomului-tată;
- al doilea cromozom-urmaș primește partea stângă a cromozomului-tată și partea dreaptă a cromozomului-mamă.

Fiecare pereche de cromozomi-părinți generează doi cromozomi-urmași și dimensiunea populației rămâne constantă.

Pasul 5 *Mutația.* Unei gene a cromozomului-urmaș i se aplică o mutație cu probabilitatea $\varepsilon \in [0.001, 0.01]$, și constă în generarea unui număr aleatoriu nr_i, între 1 și numărul de arce care ies din vârful respectiv, pentru o genă a cromozomului-urmaș. Celelalte gene rămân nemodificate.

Pasul 6 *Verificarea condiției de oprire.* Oprirea execuției algoritmului are loc dacă se respectă condiția $\left|f\left(x_{P(i)}\right) - f(x_{P(i-1)})\right| \leq \varepsilon$ pentru soluțiile asociate primului cromozom din două populații consecutive $P(i-1)$ și $P(i)$. În calitate de soluție a problemei servește soluția admisibilă care corespunde cromozomului cu valoarea minimă a funcției obiectiv din ultima populație construită. Se trece la *Pasul 2* dacă condiția de oprire nu este satisfăcută, în caz contrar **STOP**.

Remarca 2.3 *Pentru oprirea execuției algoritmului pot fi utilizate și alte condiții de oprire, spre exemplu*:

a) Sunt generate k populații;

b) Se verifică restricția de timp impusă de utilizator pentru cazul problemelor de dimensiuni foarte mari.

Exemplul (codificare) 2.1. Se consideră rețeaua de transport, din Figura 2.1, cu mulțimea de vârfuri $\{1,2,3,4\}$, unde $\{1\}$ este sursă iar $\{4\}$ este destinație, și mulțimea de arce $\{e_1, e_2, e_3, e_4, e_5, e_6\}$. Se cere de transportat fluxul de produs de 100 u. c. din sursă în destinație astfel încât costul de transport să fie minim. Un cromozom a populației, în acest caz este o mulțime de dimensiunea 4, coincide cu numărul de vârfuri. Pe fiecare poziție $i = \overline{1,4}$ este generat aleatoriu un număr care arată prin câte arce este transportat fluxul de produs, ca de exemplu:

Cromozom $\{2,2,1,0\}$.

Deci, din vârful 1 ies 2 arce, din vârful 2 ies două arce, din vârful 3 iese 1 arc și din vârful 4 nu iese nici un arc. Pentru o rețea cu 4 vârfuri se construiește o populație care conține 16 cromozomi de această formă. Aleatoriu se selectează arcele prin care este transportat flux. Să presupunem că cele 5 arce sunt $\{e_1, e_3, e_4, e_5, e_6\}$ ceea ce înseamnă că prin arcul e_2 nu este transportat flux de produs.

Cantitatea de flux care trece prin fiecare arc este descrisă de matricea proporțiilor de repartiție $R = \{r_{ij} | i = \overline{1,4}, j = \overline{1,4}\}$, cunoscând că fluxul care intră în sursă este de 100 u. c.

Fig. 2.1. Rețea cu o sursă și o destinație.
Sursa: elaborat de autor

Exemplul (decodificare) 2.2. Fie este dată matricea $R = \{\{0.0,0.45,0.0,0.55\}, \{0.0,0.0,0.30,0.70\}, \{0.0,0.0,0.0,1.0\}, \{0.0,0.0,0.0\}\}$.

Se asociază o soluție cromozomului $\{2,2,1,0\}$:

- pe arcul e_1 trece 0.45 din fluxul total de 100 u. c. care intră în vârful 1, deci $x_1 = 45$ u. c.;
- deoarece pe arcul e_2 nu trece flux atunci $x_2 = 0$ u. c.;
- pe arcul e_3 trece 0.55 din fluxul total de 100 u. c. care intră în vârful 1, deci $x_3 = 55$ u. c.;
- pe arcul e_4 trece 0.30 din fluxul total de 45 u. c. care intră în vârful 2, deci $x_4 = 15$ u. c.;
- pe arcul e_5 trece 0.70 din fluxul total de 45 u. c. care intră în vârful 2, deci $x_5 = 30$ u. c.;
- pe arcul e_6 trece 1.0 din fluxul total de 15 u. c. care intră în vârful 3, deci $x_5 = 15$ u. c.;

În așa fel, soluția admisibilă asociată cromozomului $\{2,2,1,0\}$ are forma $x = (45,0,55,15,30,15)$.

După ce este construită soluția admisibilă, se calculează valoarea funcției obiectiv pentru estimarea corectă a posibilității de transferare în noua populație. În cazul unui cromozom promițător, el este transferat în noua populație, se păstrează soluția asociată și valoarea funcției obiectiv, astfel încât pentru populația nouă acest procedeu (de determinare a soluției admisibile și de calcul al valorii funcției obiectiv respective) este realizat doar pentru urmași. Deci, odată găsită soluția admisibilă și transferat cromozomul într-o nouă populație, calculele de decodificare nu mai trebuie din nou executate, fapt care permite reducerea timpului total de execuție a algoritmului. Acest lucru este foarte important, mai ales în cazul soluționării problemelor de dimensiuni mari.

Exemplul (încrucișare) 2.3. Se consideră doi cromozomi-părinți care participă la încrucișare:

Cromozom-mama {1,1,2,1,2,1,1,1}

Cromozom-tata {1,2,1,2,3,2,1,1}

atunci cromozomii-urmași sunt:

Cromozom-urmaș1 {1,1,2,1, 3,2,1,1}

Cromozom-urmaș2 {1,2,1,2, 2,1,1,1}.

Exemplul (mutație) 2.4. Mutația presupune modificarea unei gene aleatorii a cromozomului-urmaș după cum urmează:

Cromozom-urmaș {1,1,2,1,3,2,1,1} ⇒ Cromozom-urmaș {1,1,2,1,1,2,1,1}.

Remarca 2.4. *Algoritmul AG1 se aplică în cazul rețelelor care îndeplinesc următoarele condiții:*

a) *Graful rețelei de transport este aciclic;*

b) *Fiecare vârf al grafului, cu excepția vârfului destinație, conține cel puțin un arc care iese din vârful respectiv;*

c) *Vârful destinație nu conține arce care ies din el.*

Se formulează și se demonstrează următoarele teoreme:

Teorema 2.7. *Algoritmul AG1 necesită un volum de memorie de ordinul* $O(n^2)$.

Demonstrație. *Rețeaua de transport este descrisă de graful dat de o listă de adiacență de dimensiunea* m. *Matricea cu partea de flux* $R = \{r_{ij} | i = \overline{1,n}, j = \overline{1,n}\}$ *este de dimensiunea* n^2.

Fiecare cromozom constă dintr-o listă de dimensiunea n. Astfel, o populație ce constă din $4n$ cromozomi are dimensiunea $4n^2$. Populația este înnoită la fiecare pas, deci în momentul construirii unei noi populații acesteia nu-i este alocată memorie suplimentară. Prin urmare, implementarea algoritmului AG1 necesită un volum de memorie de ordinul $O(n^2)$. □

Teorema 2.8. *Complexitatea unei iterații a algoritmului AG1 este de ordinul $O(nm)$.*

Demonstrație. *Pentru completarea listei de adiacență, care descrie graful, sunt necesare $O(m)$ operații. Procesul de generare a unui cromozom este de complexitate $O(n)$. Generarea întregii populații necesită $O(nm)$ operații. Evaluarea soluției asociate unui cromozom necesită $O(m)$ operații. Deoarece în populație sunt $4n$ cromozomi, complexitatea de evaluare a tuturor cromozomilor este $O(nm)$. Încrucișarea și mutația are complexitatea $O(n)$ pentru un cromozom și complexitatea $O(n^2)$ pentru întreaga populație. Prin urmare, o iterație a algoritmului AG1 are complexitatea de ordinul $O(nm)$.* □

Definiția 2.1. *Soluție ε-optimă a problemei neliniare de transport pe rețea este soluția $x_{P(i)}$ generată de o populație $P(i)$ care satisface condiția: $\left|f\left(x_{P(i)}\right) - f(x_{P(i-1)})\right| \leq \varepsilon$, unde $f\left(x_{P(i)}\right)$ este valoarea funcției obiectiv pentru soluția asociată cromozomului cu cele mai bune caracteristici din populația $P(i)$.*

Teorema 2.9. *Algoritmul AG1 converge către o soluție ε- optimă.*

Demonstrație. *Din populația $P(i-1)$ sunt transferați în $P(i)$ cromozomii pentru care valoarea funcției obiectiv în soluția asociată este minimă. Algoritmul converge către o soluție admisibilă care este și un minim local și satisface condiția $\left|f\left(x_{P(i)}\right) - f(x_{P(i-1)})\right| \leq \varepsilon$, ceea ce înseamnă că algoritmul întotdeauna converge către o soluție ε-optimă.* □

Observația 2.1. *Timpul de execuție a algoritmului AG1 este $O(Unm)$, unde U – numărul de iterații necesar pentru obținerea soluției ε-optimă.*

Remarca 2.5. *Pentru a evita păstrarea unui număr mare de zerouri care fac parte din matricea $R = \{r_{ij} | i = \overline{1,n}, j = \overline{1,n}\}$, implementarea practică a algoritmului presupune crearea unei liste de adiacență care conține doar ratele fluxului ce trece prin fiecare arc al rețelei ce iese din vârful i din totalul fluxului ce intră în vârful i. În așa fel, pentru păstrarea ratelor de flux pe arce este necesar un volum de ordinul $O(m)$ memorie în loc de $O(n^2)$.*

2.3.2 Implementarea și testarea algoritmului AG1

Un exemplu practic de aplicare a algoritmului AG1 pentru soluționarea problemei neliniare de transport pe rețea descrisă de o sursă și o destinație este prezentat în Anexa 5. Codul scris în Wolfram Mathematica poate fi accesat pe https://github.com/TatianaPasa/GeneticAlgorithm.

Algoritmul AG1 este implementat în limbajul Wolfram și testat în baza unui șir de exemple, rețele de diferite dimensiuni cu un număr diferit de n vârfuri și m arce ale grafului ce descriu rețeaua de transport. Testele sunt efectuate aplicând două criterii de oprire:

1) execuția algoritmului ia sfârșit atunci când se respectă condiția: $\left|f\left(x_{P(i)}\right) - f\left(x_{P(i-1)}\right)\right| \leq \varepsilon$, pentru soluțiile ce corespund valorii minime a funcției obiectiv în soluțiile $x_{P(i-1)}$ și $x_{P(i)}$ asociate cromozomilor cu cele mai bune caracteristici din două populații consecutive $P(i-1)$ și $P(i)$;
2) execuția algoritmului ia sfârșit după ce sunt generate k populații și în calitate de soluție a problemei formulate este selectată acea soluție din ultima populație căreia îi corespunde valoarea minimă a funcției obiectiv.

În Tabelul 2.4 sunt prezentate valorile timpului de execuție a problemelor de diferite dimensiuni, descrisă de numărul de vârfuri n, pentru care au fost aplicate fiecare dintre cele două criterii de oprire. Se observă că pentru cazul 1 timpul de execuție (t_ε) crește mult mai lent în funcție de numărul de vârfuri ale grafului, deoarece depinde și de numărul de iterații de care este nevoie pentru a obține o soluție ε-optimă în comparație cu cazul 2 (t_k), când algoritmul execută k iterații (în cazul dat $k = 10$) pentru probleme de diferite dimensiuni. Aceasta se datorează faptului că dacă execuția algoritmului ia sfârșit după un număr fix de iterații poate fi situația că soluția optimă obținută la iterația $p < k$, care nu mai poate fi îmbunătățită, este transmisă de la o populație la alta, deoarece condiția de oprire nu este satisfăcută. La o mutație reușită poate avea loc ieșirea din blocaj și îmbunătățirea soluției.

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

Tabelul 2.4. Timpul de execuție pentru algoritmul AG1. Sursa: elaborat de autor

n	t_ε (secunde)	t_k (secunde)	n	t_ε (secunde)	t_k (secunde)
10	0.0262	0.0950	**60**	6.3544	46.7701
15	0.0809	0.3066	**65**	4.9332	83.5933
20	0.2421	0.7972	**70**	3.5041	102.1030
25	0.3294	1.6901	**75**	7.4183	195.1190

30	0.5540	3.7000	**80**	18.0235	193.3200
35	1.1752	7.1764	**85**	12.8842	323.9760
40	0.9351	9.0325	**90**	10.5738	311.3430
45	1.1236	18.3405	**95**	45.6272	511.6740
50	3.5589	24.2091	**100**	55.9386	529.4140
55	2.2681	46.7701	**120**	117.9100	1621.2100

Utilizarea funcției standard $Minimize[\{f, cons\}, \{x, y, ...\}]$ a Sistemului Mathematica permite obținerea minimumului global pentru problema formulată.

Timpul de execuție a funcției *Minimize[],* dat în Tabelul 2.5, crește brusc pentru grafuri cu 8 sau mai multe vârfuri și este chiar de 3 ori mai mare decât în cazul grafului cu 100 de vârfuri pentru care este aplicat algoritmul AG1.

Tabelul 2.5. Timpul de execuție pentru funcția Minimize[]. Sursa: elaborat de autor

n	**Minimize[]**(secunde)
4	0.17
6	3.08
8	345.61

În Tabelul 2.6 pot fi vizualizate rezultatele soluționării unui șir de probleme de transport pe rețea cu n vârfuri și m arce pentru care se cunosc funcțiile concave secvențial-liniare de cost ce depind de fluxul transportat pe fiecare arc.

Tabelul 2.6. Modificarea valorii $F_T(x)$ pentru algoritmul AG1. Sursa: elaborat de autor

Iterația	**n/m** (u. c.)	**n/m** (u. c.)	**n/m** (u. c.)	**n/m** (u. c.)	**n/m** (u. c.)
	10/31	***20/94***	***30/238***	***100/2392***	***150/5875***
I	5565.54	25670.50	51252.20	359995	582184
II	4386.53	22407.90	41771.90	330423	546030
III	2999.87	19529.90	35173.60	309817	512836
IV	-	-	29722.00	-	483577
V	-	-	-	-	453874
VI	-	-	-	-	416877
F(x)	**14.00**	**106.611**	**116.00**	**357.063**	**173.952**

Datele prezentate în acest tabel denotă o micșorare a valorii funcției obiectiv totale $F_T(x)$ de la o iterație la alta, ceea ce presupune că la trecerea de la populația $P(i-1)$ la populația $P(i)$

se obțin cromozomi cu caracteristici mai bune cărora le este asociată câte o soluție pentru care valoarea funcției obiectiv $F(x)$ este mai mică.

2.3.3 Algoritmul genetic AG2

În cele ce urmează este soluționată problema neliniară de transport (2.1) – (2.3), care presupune determinarea costului minim de transport, formulată în 2.1 pentru rețele standard cu funcția de producere și consum de forma (2.8), când se impune restricția transportării unui flux de produs prin fiecare dintre arcele rețelei unde v_0 - sursa rețelei, v_t – destinația rețelei. O astfel de problemă reprezintă modelul matematic al problemei reale de transport al unui flux prin rețelele de apă, gaz sau electricitate.

În algoritmul ce urmează, fiecare genă a unui cromozom este descrisă de o matrice care conține partea de flux ce trece prin arcul care iese din vârful i, din totalul de flux ce intră în acel vârf. Pentru a economisi memoria calculatorului în timpul implementării algoritmului, fiecare cromozom este de dimensiunea m. Fiecare genă i din cromozom este descrisă de o listă care conține partea de flux a arcelor care ies din vârful i.

Operatorii de încrucișare și mutație aplicați asupra cromozomilor sunt definiți astfel încât după decodificare să fie asociată o soluție admisibilă. Pentru evitarea pierderii soluțiilor care corespund cromozomilor promițători ca rezultat al operațiilor de selecție, încrucișare sau mutație este aplicat modelul elitist al algoritmului genetic care presupune păstrarea acestor cromozomi.

Algoritmul genetic AG2 propus pentru soluționarea problemei de transport cu funcții concave nedescrescătoare secvențial-liniare constă din următorii pași:

Pasul 1 *Inițializarea.* Se generează populația inițială din $4n$ cromozomi: fiecare cromozom este descris de o listă de forma $L = \left\{\{l_{i1}, \dots, l_{im_i}\}_i \mid i = \overline{1, n-1}\right\}$, unde m_i – numărul de arce care ies din vârful i, fiecare $l_{ij}, i = \overline{1, n-1}, j = \overline{1, m_i}$ reprezentând partea de flux care trece prin arcul (i, j) din totalul fluxului care intră în vârful i. Pentru fiecare vârf se cere să fie satisfăcută condiția $\sum_{j=1}^{m_i} l_{ij} = 1, i = \overline{1, n-1}$.

Pasul 2 *Decodificarea și Evaluarea cromozomilor.* Decodificarea presupune asocierea fiecărui cromozom cu o soluție admisibilă de forma $x = (x_1, x_2, \dots, x_m)$, unde x_h, $h = \overline{1, m}$ este cantitatea de flux care trece prin arcul (i, j) și se determină astfel $x_h = l_{ij} * cf_i$, $i = \overline{1, n-1}$, $j = \overline{1, m_i}$, unde cf_i este cantitatea de flux care intră în vârful i și l_{ij}, $i = \overline{1, n-1}$, $j = \overline{1, m_i}$ - partea de flux care trece prin arcul (i, j).

Evaluarea presupune determinarea valorii funcției obiectiv pentru fiecare dintre soluțiile obținute.

Pasul 3 *Selectarea.* Cromozomii sunt sortați în ordinea creșterii valorii funcției obiectiv în soluția asociată cromozomului. Cromozomii din prima jumătate a populației, adică $2n$ cromozomi, sunt transferați în noua populație.

Pasul 4 *Încrucișarea cromozomilor.* Încrucișarea are loc între cromozomii transferați în populația $P(i)$ din populația $P(i-1)$. Aleatoriu este aplicată aceeași tăietură ambilor cromozomi-părinți, luați câte doi, care presupune împărțirea în două mulțimi de gene a cromozomului L_1 și L_2, astfel încât $L = L_1 \cup L_2$, unde $L_1 = \left\{\left\{l_{i1}, \dots, l_{im_i}\right\}_i \mid i = \overline{1, n_1}\right\}$, $L_2 = \left\{\left\{l_{i1}, \dots, l_{im_i}\right\}_i \mid i = \overline{n_1+1, n}\right\}$. L_1 – partea stângă iar L_2 - partea dreaptă a cromozomului.

Încrucișarea cromozomilor presupune:

- primul cromozom-urmaș primește partea stângă a cromozomului-mamă și partea dreaptă a cromozomului-tată;
- al doilea cromozom-urmaș primește partea stângă a cromozomului-tată și partea dreaptă a cromozomului-mamă.

Fiecare pereche de cromozomi-părinți generează doi cromozomi-urmași și dimensiunea populației generate rămâne constantă.

Pasul 5 *Mutația.* Unei gene a cromozomului i se aplică mutația cu o probabilitate $\varepsilon \in [0.1, 0.5]$ și presupune modificarea unei gene de pe poziția i a cromozomului-urmaș. Se generează o listă nouă $\left\{l_{i1}, \dots, l_{im_i}\right\}_i$ pentru un vârf i ales aleatoriu, astfel încât să fie satisfăcută condiția $\sum_{j=1}^{m_i} l_{ij} = 1, i = \overline{1, n-1}$. Celelalte gene rămân nemodificate.

Pasul 6 *Verificarea condiției de oprire.* Oprirea algoritmului are loc dacă se respectă condiția $\left|f\left(x_{P(i)}\right) - f\left(x_{P(i-1)}\right)\right| \leq \varepsilon$, pentru soluțiile asociate primului cromozom din două populații consecutive $P(i-1)$ și $P(i)$. În calitate de soluție a problemei servește cea care corespunde cromozomului cu valoarea minimă a funcției obiectiv din ultima populație construită. Se trece la *Pasul 2* dacă condiția de oprire nu este satisfăcută.

Remarca 2.6. *Pentru oprirea algoritmului poate fi utilizată și una din următoarele condiții*:

a) Sunt generate k populații;

b) Se verifică restricția de timp, impusă de utilizator, pentru cazul problemelor de dimensiuni foarte mari.

Exemplul (codificare) 2.5. Se consideră rețeaua de transport, din Figura 2.2, cu mulțimea de vârfuri $\{1,2,3,4,5\}$, unde $\{1\}$ este sursă iar $\{5\}$ este destinație, și mulțimea de arce $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$. Se cere de transportat fluxul de produs de 100 u. c. din sursă în destinație astfel încât să fie repartizat prin toate arcele rețelei și costul de transport să fie minim. Un cromozom a populației este o mulțime de dimensiunea 9 care coincide cu numărul de arce.

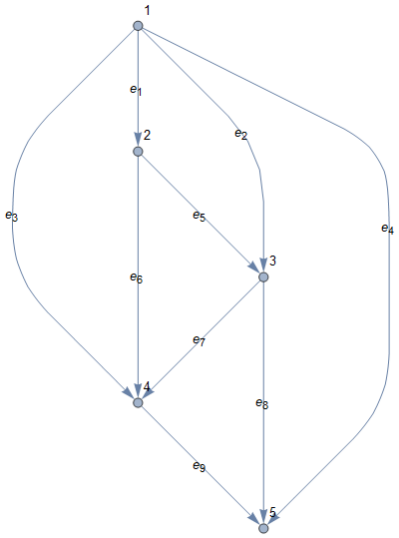


Fig. 2.2. Rețea cu o sursă și o destinație cu flux prin toate arcele. Sursa: elaborat de autor

Astfel, un cromozom are forma generală: $L = \left\{\{l_{e_1}, l_{e_2}, l_{e_3}, l_{e_4}\}_1, \{l_{e_5}, l_{e_6}\}_2, \{l_{e_5}, l_{e_8}\}_3, \{l_{e_1}\}_4\right\}$, care descrie:

- $\{l_{e_1}, l_{e_2}, l_{e_3}, l_{e_4}\}_1$ - partea de flux care trece prin arcele e_1, e_2, e_3, e_4 din totalul de flux ce intră în vârful 1;
- $\{l_{e_5}, l_{e_6}\}_2$ - partea de flux care trece prin arcele e_5, e_6 din totalul de flux ce intră în vârful 2;
- $\{l_{e_7}, l_{e_8}\}_3$ - partea de flux care trece prin arcele e_7, e_8 din totalul de flux ce intră în vârful 3;
- $\{l_{e_9}\}_4$ - partea de flux care trece prin arcul e_9, deoarece din vârful 4 iese doar un arc;
- din vârful 5 nu iese nici un arc.

Pentru rețeaua din Figura 2.2, un cromozom, care respectă și condiția că tot fluxul care intră într-un vârf este repartizat pe arcele care ies din el, are forma:

$$\{\{0.30, 0.44, 0.10, 0.16\}_1, \{0.70, 0.30\}_2, \{0.40, 0.60\}_3, \{1.0\}_4\}.$$

Exemplul (decodificare) 2.6. Pentru cromozomul construit mai sus se asociază o soluție admisibilă de forma $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9)$ după cum urmează:

- $x_1 = 0.30 * 100 = 30$ u. c. de flux sunt transportate pe arcul e_1;
- $x_2 = 0.44 * 100 = 44$ u. c. de flux sunt transportate pe arcul e_2;
- $x_3 = 0.10 * 100 = 10$ u. c. de flux sunt transportate pe arcul e_3;
- $x_4 = 0.16 * 100 = 16$ u. c. de flux sunt transportate pe arcul e_4;
- $x_5 = 0.70 * 30 = 21$ u. c. de flux sunt transportate pe arcul e_5;
- $x_6 = 0.30 * 30 = 9$ u. c. de flux sunt transportate pe arcul e_6;
- $x_7 = 0.40 * 65 = 26$ u. c. de flux sunt transportate pe arcul e_7;
- $x_8 = 0.60 * 65 = 39$ u. c. de flux sunt transportate pe arcul e_8;
- $x_9 = 1.0 * 45 = 45$ u. c. de flux sunt transportate pe arcul e_9;

Deci, soluția admisibilă obținută este: $x = (30, 44, 10, 16, 21, 9, 26, 39, 45)$.

După construirea soluției admisibile se calculează valoarea funcției obiectiv în această soluție pentru estimarea corectă a posibilității de transferare în noua populație. În cazul unui cromozom promițător, el este transferat în noua populație fiind păstrată soluția admisibilă asociată și valoarea funcției obiectiv, astfel încât pentru populația nouă acest procedeu (de determinare a soluției admisibile și de calcul al valorii funcției obiectiv respective) se realizează doar pentru cromozomii-urmași.

Exemplul (încrucișare) 2.7. Se consideră doi cromozomi-părinți care participă la încrucișare:

Cromozom-mama {{0.16,0.35,0.07,0.42},{0.34,0.66},{0.28,0.72},{1.}}

Cromozom-tata {{0.19,0.32,0.11,0.38},{0.81,0.19},{0.67,0.33},{1.}}

atunci cromozomii-urmașii sunt:

Cromozom-urmaș1 {{0.16,0.35,0.07,0.42},{0.34,0.66},{0.67,0.33},{1.}}

Cromozom-urmaș2 {{0.19,0.32,0.11,0.38},{0.81,0.19},{0.28,0.72},{1.}}.

Exemplul (mutație) 2.8. Mutația presupune modificarea unei gene aleatorii a cromozomului-urmaș. Pentru următorul:

Cromozom-urmaș {{0.16,0.35,0.17,0.32},{0.81,0.19},{0.67,0.33},{1.}}

după mutație se obține:

Cromozom-urmaș {{0.16,0.35,0.17,0.32},{0.26,0.74},{0.67,0.33},{1.}}.

Remarca 2.7. *Algoritmul AG2 se aplică în cazul rețelelor care îndeplinesc următoarele condiții:*

a) *Graful rețelei de transport este aciclic;*

b) *Fiecare vârf al grafului, cu excepția vârfului destinație, conține cel puțin un arc care iese din vârful respectiv;*

c) *Vârful destinație nu conține arce care ies din el.*

Se formulează și se demonstrează următoarele teoreme:

Teorema 2.10. *Algoritmul AG2 necesită un volum de memorie de ordinul* $O(nm)$.

Demonstrație. *Rețeaua de transport este descrisă de graful dat de o listă de adiacență de dimensiunea* m. *Fiecare cromozom constă dintr-o listă* L *de dimensiunea* m. *Astfel, o populație*

ce constă din $4n$ cromozomi are dimensiunea $4nm$. Populația este înnoită la fiecare pas, deci în momentul construirii unei noi populații acesteia nu-i este alocată memorie suplimentară. Prin urmare, implementarea algoritmului AG2 necesită un volum de memorie de ordinul $O(nm)$.□

Teorema 2.11. *Complexitatea unei iterații a algoritmului AG2 este de ordinul $O(nm)$.*

Demonstrație. *Pentru completarea listei de adiacență care descrie graful sunt necesare $O(m)$ operații. Procesul de generare a unui cromozom are complexitate $O(m)$. Generarea întregii populații necesită $O(nm)$ operații. Evaluarea soluției asociate unui cromozom necesită $O(m)$ operații. Deoarece în populație sunt $4n$ cromozomi, complexitatea de evaluare a tuturor cromozomilor este $O(nm)$. Încrucișarea și mutația are complexitatea $O(m)$ pentru un cromozom și complexitatea $O(nm)$ pentru întreaga populație. Prin urmare, o iterație a algoritmului AG2 are complexitatea de ordinul $O(nm)$.* □

Observația 2.2. *Din Teorema 2.11 rezultă că timpul de execuție a algoritmului AG2 este $O(Unm)$, unde U este numărul de iterații necesar pentru obținerea unei soluții ε-optime.*

Teorema 2.12. *Algoritmul AG2 converge către o soluție ε-optimă.*

Demonstrație. *Din populația $P(i-1)$ sunt transferați în $P(i)$ cromozomii pentru care valoarea funcției obiectiv în soluția asociată este minimă. Algoritmul converge către o soluție admisibilă care este și un minim local și satisface condiția $\left|f\left(x_{P(i)}\right) - f(x_{P(i-1)})\right| \leq \varepsilon$, ceea ce înseamnă că algoritmul converge către o soluție ε-optimă.*□

Deși problema formulată presupune determinarea unui flux de cost minim într-o rețea de transport cu funcții concave de cost și condiția că fluxul acoperă întreaga rețea, poate fi și situația când cantitatea de flux transportat pe un arc să fie mărginită de jos și de sus, adică există mărimile pozitive corespunzătoare $l_j, j = \overline{1,m}$ și $u_j, j = \overline{1,m}$, încât: $l_j < x_j \leq u_j, j = \overline{1,m}$. Aici $x = (x_1, x_2, \dots, x_m)$ este flux admisibil în rețea. Se presupune că restricțiile permit transportarea întregului flux din sursă în destinație. În acest context, algoritmul AG2 poate fi adaptat pentru problema de transport de cost minim pe rețea cu restricții asupra capacității inferioare și superioare a arcelor.

Remarca 2.8. *În cazul în care sunt impuse restricții inferioare și superioare a capacităților arcelor pentru transportul fluxului de produs astfel încât costul este minim, se ține cont de următoarele observații:*

a) La generarea populației inițiale, fiecare cromozom este decodificat și se verifică dacă soluția ce îi corespunde satisface restricțiile unei soluții admisibile și numai apoi se adaugă în populație;

b) La încrucișarea cromozomilor, în caz că se obține o soluție care nu satisface restricțiile unei soluții admisibile, cromozomul-urmaș este înlocuit cu cromozomul-părinte ce corespunde soluției pentru care valoarea funcției obiectiv este minimă;

c) La mutația unei gene, în cazul nerespectării restricțiilor unei soluții admisibile asociate cromozomului obținut, mutația este repetată de mai multe ori până la îndeplinirea restricțiilor sau poate fi anulată (după caz).

2.3.4 Implementarea și testarea algoritmului AG2

Un exemplu practic de aplicare a algoritmului AG2 pentru soluționarea problemei neliniare de transport pe rețea descrisă de o sursă și o destinație cu condiția că fluxul este transportat prin toate arcele rețelei este prezentat în Anexa 6. Codul scris în Wolfram Mathematica poate fi accesat pe https://github.com/TatianaPasa/GeneticAlgorithm.

Algoritmul AG2 este realizat în limbajul Wolfram și testat în baza unui șir de exemple, rețele de transport de variate dimensiuni cu un număr diferit de vârfuri n și de arce m ale grafului ce descriu rețeaua. Testele sunt efectuate aplicând două criterii de oprire:

1) execuția algoritmului ia sfârșit atunci când se respectă condiția $|f(i-1)-f(i)| \leq \varepsilon$, pentru soluțiile ce corespund valorii funcției obiectiv minime din două populații consecutive $P(i-1)$ și $P(i)$;
2) execuția algoritmului ia sfârșit după ce sunt generate k populații și în calitate de soluție a problemei formulate este selectată acea soluție din ultima populație căreia îi corespunde valoarea minimă a funcției obiectiv.

Din Tabelul 2.7 se poate observa că pentru cazul 2 timpul de execuție (t_k) crește odată cu creșterea dimensiunii rețelei de transport (sunt realizate 10 iterații). Aceeași tendință nu este observată pentru cazul 1 cu t_ε, care depinde nu doar de dimensiunea rețelei de transport, dar și de numărul de iterații care sunt necesare pentru obținerea unei soluții ε-optime. Ca și pentru algoritmul AG1, criteriul 2 poate fi aplicat cu succes când nu se cere obținerea unei soluții ε-optime dar este destul să se determine un flux de cost minim care curge din sursă la destinație fiind repartizat prin toate arcele rețelei.

Tabelul 2.7. Timpul de execuție pentru algoritmul AG2. Sursa: elaborat de autor

n / m	$\mathbf{t_\varepsilon}$ (secunde)	t_k (secunde)	**n / m**	$\mathbf{t_\varepsilon}$ (secunde)	t_k (secunde)
10 / 33	0.0781	0.1875	**50 / 647**	9.9687	27.7813
15 / 62	0.3125	0.2300	**65 / 1236**	56.4183	70.9531
20 / 105	0.4843	1.7675	**70 / 1274**	42.9375	78.9219
25 / 168	0.9687	3.4843	**80 / 1764**	45.7031	130.0320
30 / 213	4.8593	5.4218	**90 / 2275**	216.6410	194.5470
35 / 321	5.0781	9.4218	**100 / 2572**	354.4537	244.5780
40 / 402	7.3593	13.4219	**120 / 3633**	237.4530	430.5310

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

În Tabelul 2.8 sunt incluse rezultatele soluționării unui șir de probleme de transport pe rețea cu n vârfuri și m arce pentru care se cunosc funcțiile concave de cost ce depind de fluxul transportat pe fiecare arc și funcția de producere și consum.

Practic este demonstrată convergența algoritmului. Fiind calculată valoarea funcției obiectiv totale a populației se observă o descreștere continuă a acestei mărimi.

Tabelul 2.8. Modificarea valorii $F_T(x)$ pentru algoritmul AG2. Sursa: elaborat de autor

Iterația	**n/m** (u. c.)	**n/m** (u. c.)	**n/m** (u. c.)	**n/m** (u. c.)	**n/m** (u. c.)
	10/31	***20/94***	***30/238***	***100/2392***	***150/5875***
I	8352.59	34161	74520	479731	683430
II	7698.17	31704.1	70750	463270	658387
III	7246.51	-	68168	461585	640905
IV	-	-	65941	454880	624014
V	-	-	63955	448784	608769
VI	-	-	61794	442621	593624
VII	-	-	59992	437814	579315
VIII	-	-	58840	433995	564279
IX	-	-	57422	430823	550445
X	-	-	56276	428015	535530
XI	-	-	56276	-	525713
F(x)	**160**	**270**	**430**	**1000**	**820**

Datele prezentate în tabel atestă o micșorare a $F_T(x)$ de la o iterație la alta, ceea ce presupune că la trecerea de la o populație $P(i-1)$ la o populație $P(i)$ este o îmbunătățire a soluției problemei.

2.4 Soluționarea problemei neliniare de transport pe rețea cu o sursă și câteva destinații utilizând algoritmi genetici

Deseori este întâlnită situația când un mare producător al produsului X are contracte de vânzare cu comercianți din diferite orașe sau/și țări. La rândul lor comercianții au mai multe depozite în diferite orașe sau/și câteva în același oraș de unde se face livrare la micii antreprenori. Într-un final, după ce produsul a trecut prin una sau mai multe puncte intermediare, el este plasat pe rafturile magazinelor de unde este procurat de cumpărător. Produsul este transportat de la un producător care este sursa rețelei de transport și mai departe dintr-un punct în altul, numite puncte intermediare, până ajunge în magazine care sunt destinațiile rețelei de transport. Această structură poate fi descrisă de o rețea de transport cu o sursă, câteva destinații și cu o mulțime de puncte intermediare. În sursă este producere, în destinații este consumat tot ce s-a produs, iar în punctele intermediare nu este nici producere și nici consum de produs. Respectarea acestor condiții implică existența unui flux în rețeaua de transport descrisă, pentru care se cere minimizarea costului de transport a produsului din sursă în destinații.

O modalitate de soluționare a problemei este conectarea tuturor destinațiilor la o super-destinație asociind arcelor ce le unește câte o funcție de cost nulă. În acest caz, cel mai probabil soluția nu permite aprovizionarea tuturor destinațiilor cu necesarul dorit deoarece problema astfel reformulară nu conține restricțiile respective, scopul fiind ca produsul pornit din sursă să ajungă în super-destinație cu un cost minim de transport. Deci, problema soluționată nu coincide cu problema inițial formulată care constă în aprovizionarea tuturor destinațiilor astfel încât costul de transport să fie minim.

Pentru soluționarea problemelor de dimensiuni mari nu se cunosc metode care ar furniza soluția optimă în timp polinomial. Aceasta din cauza specificului problemei NP-dificile cu funcții concave de cost care poate avea *mai multe minime locale*, spre care, în cele mai dese cazuri, converge algoritmul.

Algoritmii prezentați în (Lozovanu & Pasha, 2002), (Pașa & Ungureanu, 2017a), (Pașa & Ungureanu, 2017d), (Pașa & Ungureanu, 2017e), (Pașa & Ungureanu, 2017f), propuși și analizați în secțiunea 2.2, deși se execută mai repede decât funcțiile standard din Sistemul Mathematica bazate pe metode clasice de soluționare (Pașa, 2018a), când vorbim de rețele de transport de dimensiuni mari, timpul de execuție întrece timpul rezonabil.

În astfel de situații sunt recomandați algoritmii genetici care nu necesită cunoașterea gradienților și a Hesianului. Algoritmii genetici mențin un echilibru între exploatarea și explorarea mulțimii de soluții admisibile (Holland, 1975), (Hoopgood, 2012). Încorporarea tehnicilor ce permit îmbunătățirea soluției algoritmului genetic implică hibrid (Sheng, Dechen, & Xiaofei, 2006) care livrează soluții mai bune.

Elitismul (Kenneth, 1975), implementat în algoritmii genetici AG1 – AG2 expuși mai sus, permite păstrarea cromozomilor cu cele mai bune caracteristici la trecerea de la o populație la alta, în caz contrar cromozomii ar putea fi pierduți prin neselectare sau prin distrugerea lor când sunt aplicați operatorii de încrucișare sau mutație. Același model este aplicat și în cazul algoritmului genetic descris mai jos.

2.4.1 Formularea problemei

În cele ce urmează este soluționată problema neliniară de transport (2.1) – (2.3), formulată în secțiunea 2.1, pe o rețea de transport cu o singură sursă și câteva destinații descrisă de mulțimea de vârfuri V, unde $|V| = n$ și mulțimea de arce E, unde $|E| = m$. Problema presupune determinarea costului minim de transport. Funcția de producere și consum asociată mulțimii de vârfuri are forma:

$$q(v) = 0 \text{ pentru } v \in (V \backslash V_t) \backslash \{v_0\}$$
$$q(v_0) = \sum_{i=1}^{k} q(v_i), v_i \in V_t \tag{2.9}$$

unde v_0 – sursa rețelei, $v_i \in V_t, i = \overline{1,k}$ – destinația i a rețelei de transport, $V_t \subset V$ – mulțimea tuturor destinațiilor rețelei și $|V_t| = k$.

După cum este menționat și în secțiunea 2.1, problema neliniară de transport pe rețea (2.1) – (2.3), (2.9) cu funcții concave secvențial-liniare de cost poate fi soluționată utilizând algoritmi finiți ce se bazează pe cercetarea tuturor grafurilor fără cicluri. Fiecăruia îi este asociată o soluție admisibilă a problemei. Arborele de acoperire pentru un graf ce descrie rețeaua de transport satisface condițiile unei soluții admisibile a problemei formulate, ceea ce implică cercetarea tuturor soluțiilor admisibile construite pe arborii de acoperire și determinarea acelui care descrie soluția optimă.

Se calculează valoarea funcției obiectiv ce descrie costul de transport pentru fiecare soluție astfel construită și este acceptată cea care corespunde funcției de cost minim.

În cazul problemelor descrise de grafuri dense și/sau de dimensiuni mari, pentru soluționarea cărora verificarea tuturor soluțiilor, asociate cu arborii de acoperire, este imposibilă

din cauza timpului sau a tehnicii care nu permite prelucrarea unor date atât de voluminoase, se recomandă utilizarea algoritmilor genetici. Astfel de algoritmi permit păstrarea candidaților promițători, eliminarea pretendenților cărora le sunt asociate costuri mari de transport și, astfel, micșorarea semnificativă a numărului de soluții admisibile care trebuie cercetate pentru a obține un rezultat bun în timp rezonabil.

2.4.2 Algoritmul genetic AG3

Selectarea permite aplicarea căutării locale a soluției, astfel încât cromozomii cu caracteristici bune ai populației curente sunt selectați și transferați în noua populație pentru a participa la crearea noilor cromozomi.

Aplicarea operatorilor de *încrucișare* și *mutație*, în algoritmul AG3 descris mai jos, poate conduce la situații în care nu sunt respectate restricțiile de existență a fluxului în rețea, deci nu poate fi asociată o soluție admisibilă aplicând decodificarea unui astfel de cromozom. În astfel de situații sunt utilizate metode din teoria grafurilor, de adaptare a cromozomilor, care permit să fie aduși la o formă astfel încât după decodificarea lor să se obțină soluții admisibile. Încrucișarea constă în transmiterea unor gene de la un cromozom-părinte și a altor gene de la alt cromozom-părinte iar mutația presupune modificarea unei gene selectată aleatoriu.

Dimensiunea populației și modalitatea de codificare a problemei este un element important care determină cantitatea memoriei necesare pentru păstrarea datelor. Fiecare cromozom permite păstrarea doar a arcelor care fac parte din arborele de acoperire, astfel micșorând dimensiunea cromozomului de la m la n, precum și viteza de execuție a unei iterații. În același timp, dimensiunea populației trebuie să permită prelucrarea unui număr suficient de soluții admisibile a problemei astfel încât să fie depistată cea optimă în timp rezonabil.

Codificarea unei soluții admisibile presupune asocierea cu un arbore de acoperire a grafului ce descrie rețeaua de transport. Fiecare cromozom, care descrie o soluție admisibilă, conține $n-1$ gene date de perechi de numere (i,j), care indică arcul ce face parte din arbore, pornește din vârful i și intră în vârful j.

Algoritmul genetic AG3 propus pentru soluționarea problemei de transport cu funcții concave nedescrescătoare secvențial-liniare constă din următorii pași:

Pasul 1 *Inițializarea.* Se generează aleatoriu populația inițială din $4n$ cromozomi: fiecare cromozom este descris de o listă de forma $T = \{(i,j) \mid i,j = \overline{1,n}\}$, unde T este arbore de acoperire a grafului $G(V,E)$, $|T| = n-1$ și fiecare (i,j) este arcul ce iese din vârful i și

intră în vârful j. În cazul prezenței repetărilor aceluiași arbore de acoperire, repetările sunt șterse și dimensiunea populației este adaptată (a se vedea Remarca 2.9).

Pasul 2 *Decodificarea și Evaluarea cromozomilor.* Decodificarea presupune că fiecărui cromozom i se asociază o soluție admisibilă. În acest scop:

1. se asociază flux nul arcelor din graful inițial, care nu aparțin arborelui de acoperire ce descrie cromozomul;
2. se asociază flux arcelor care intră în destinații cantitatea căruia coincide cu necesarul destinației respective;
3. parcurgând arborele în adâncime, arcelor rămase (i, j) le este asociat flux cantitatea căruia coincide cu totalul fluxului care trece prin arcele descendente vârfului j.

Pentru soluția admisibilă de forma $x = (x_1, x_2, \dots, x_m)$, unde x_h, $h = \overline{1,m}$ este cantitatea de flux care trece prin arcul (i, j) cu indicele h avem:

$$x_h = \begin{cases} 0, & (i,j) \notin T, \\ q(j), & j \in V_t, \\ \sum_{(j,j')} x_{r'}, & r' \text{ este indicele arcului } (j,j'). \end{cases}$$

Evaluarea presupune determinarea valorii funcției obiectiv pentru fiecare dintre soluțiile obținute.

Pasul 3 *Selectarea cromozomilor.* Cromozomii sunt sortați în ordinea creșterii valorii funcției obiectiv asociate cromozomului. Cromozomii din prima jumătate a populației sunt transferați în noua populație.

Pasul 4 *Încrucișarea cromozomilor.* Are loc între cromozomii transferați în populația $P(i)$ din populația $P(i-1)$ prin selecție.

Aleatoriu se aplică aceeași tăietură ambilor cromozomi-părinți luați câte doi, care constă în împărțirea aleatorie a mulțimii de destinații în două submulțimi. Mulțimea de destinații $V_t = \{1, 2, \dots, k\}$ este divizată în două submulțimi $V_{t_1} = \{v_1, v_2, \dots, v_{k_1}\}$ și $V_{t_2} = \{v_{k_1+1}, v_{k_1+2}, \dots, v_k\}$, astfel încât $V_t = V_{t_1} \cup V_{t_2}$ cu $|V_t| = |V_{t_1}| + |V_{t_2}|$.

Încrucișarea cromozomilor presupune:

- primul cromozom-urmaș primește arcele care formează drumurile din sursă către mulțimea V_{t_1} a cromozomului-mamă și arcele care formează drumurile din sursă în mulțimea V_{t_2} a cromozomului-tată;
- al doilea cromozom-urmaș primește arcele care formează drumurile din sursă către mulțimea V_{t_1} a cromozomului-tată și arcele care formează drumurile din sursă în mulțimea V_{t_2} a cromozomului-mamă.

La fiecare cromozom-urmaș se adaugă vârfurile lipsă din totalul de n vârfuri ale grafului inițial și arcurile ce formează drumurile care unesc aceste vârfuri cu sursa în arborii care descriu cromozomii-părinți astfel:

- primului cromozom-urmaș i se adaugă drumurile de la cromozomul-mamă;
- celui de al doilea cromozom-urmaș i se adaogă drumurile de la cromozomul-tată.

Pentru a putea asocia cromozomul-urmaș cu o soluție admisibilă este construit un arbore de acoperire peste graful obținut care descrie cromozomul-urmaș. Fiecare pereche de cromozomi-părinți generează doi cromozomi-urmași și dimensiunea populației rămâne constantă.

Pasul 5 *Mutația.* Unei gene a cromozomului-urmaș i se aplică o mutație care are loc cu probabilitatea $\varepsilon \in [0.1, 0.5]$ și presupune ștergerea unui arc (i, j) și adăugarea altui arc (i^*, j) prezent în graful inițial, celelalte gene rămân nemodificate.

Pasul 6 *Verificarea condiției de oprire.* Implică oprirea algoritmului după k iterații. În calitate de soluție a problemei servește cea care corespunde cromozomului cu valoarea minimă a funcției obiectiv din ultima populație construită. Se trece la *Pasul 2* dacă condiția de oprire nu este satisfăcută.

Remarca 2.9. Deoarece la generarea aleatorie a arborilor, la pasul de inițializare, pot avea loc repetări, aceștia sunt șterși din populație pentru ca algoritmul să nu fie forțat să tindă către un minim local și nu spre cel global. După eliminarea repetărilor, dimensiunea populației se reduce până la cel mai mare număr de forma 4p, adică la un număr divizibil cu 4. Cu dimensiunea populației astfel obținută se operează în continuare în pașii 2-6 pentru determinarea soluției problemei considerate.

Știind cantitatea de flux necesară fiecărei destinații, se aplică procedeul de căutare în adâncime (DFS) pentru a afla soluția asociată cromozomului. Fluxul care trebuie să ajungă în destinații este repartizat din vârfuri astfel ca să se respecte legea conservării fluxului – cantitatea fluxului care intră într-un vârf intermediar este egală cu cantitatea fluxului care iese din același vârf. Astfel, cantitatea fluxului total care ajunge în destinații coincide cu cantitatea fluxului care pornește din sursă.

Exemplul (codificare) 2.9. Se consideră rețeaua de transport, din Figura 2.3, descrisă de graful cu mulțimea de vârfuri $\{1, 2,3,4,5\}$, unde $\{1\}$ este sursă iar $\{4, 5\}$ sunt destinațiile, și mulțimea de arce $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$. Se cere de transportat în destinația 4 fluxul de produs de 45 u. c. și în destinația 5 fluxul de produs de 55 u. c. din sursă care conține 100 u. c. astfel încât

costul de transport să fie minim. Un cromozom a populației, în acest caz este o mulțime de arce de dimensiunea 4 care descrie un arbore de acoperire pe graf.

Fie, se construiește un arbore de acoperire care conține mulțimea de arce $\{e_1, e_4, e_5, e_7\}$. În așa caz, un cromozom a populației este: $T = \{(1,2), (2,4), (2,3), (3,5)\}$, ceea ce înseamnă că pe arcele acestui arbore este transportat flux iar pe arcele e_2, e_3, e_6 – nu, deci este asociat un flux nul.

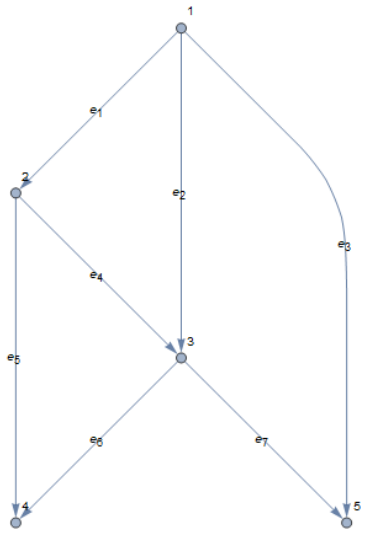


Fig. 2.3. Rețea cu o sursă și două destinații.

Sursa: elaborat de autor

Exemplul (decodificare) 2.10. Pentru cromozomul generat $T = \{(1,2), (2,4), (2,3), (3,5)\}$ se construiește soluția admisibilă de forma $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ astfel:

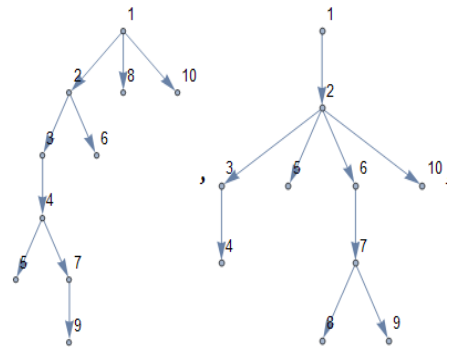
1. inițial se asociază flux nul arcelor care nu fac parte din arborele de acoperire: $x_2 = 0$ u. c., $x_3 = 0$ u.c., $x_6 = 0$ u. c.;
2. se asociază flux arcelor care intră în destinații, dar nu li s-a asociat deja flux nul: $x_5 = 45$ u. c., $x_7 = 55$ u. c.;
3. se asociază flux arcelor rămase: $x_4 = 0 + 55 = 55$ u. c., $x_1 = 45 + 55 = 100$ u. c.

Soluția admisibilă obținută este: $x = (100,0,0,55,45,0,55)$. Cunoscând soluția admisibilă se calculează costul de transport pe fiecare arc și valoarea funcției obiectiv, adică costul total de transport al fluxului din sursă către toate destinațiile rețelei.

Exemplul (încrucișare) 2.11. În Figura 2.4 a) și b) sunt prezentați doi arbori de acoperire care descriu doi cromozomi-părinți care participă la încrucișare:

Cromozom-mama $\{(1,2), (2,3), (3,4), (4,7), (7,9), (1,8), (1,10), (2,6), (4,5)\}$

Cromozom-tata $\{(1,2), (2,6), (6,7), (7,8), (7,9), (2,10), (2,3), (3,4), (2,5)\}$.



a) b)

Fig. 2.4. Cromozomi-părinți.

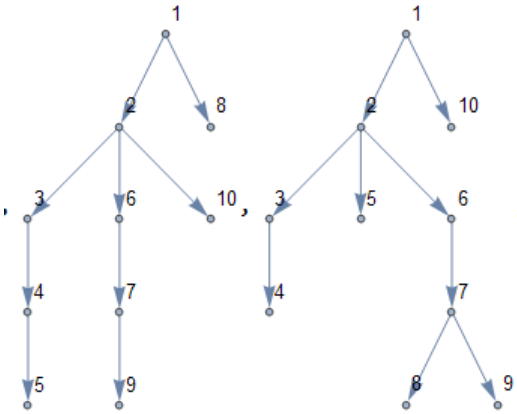
Sursa: elaborat de autor

Încrucișarea presupune aplicarea aceleiași tăieturi ambilor cromozomi-părinți. Ea constă în împărțirea aleatorie a mulțimii destinațiilor în două mulțimi, spre exemplu:

$$V_{t_1} = \{8\} \text{ și } V_{t_2} = \{9,10\}.$$

Drumul până în destinația *8* din cromozomul-părinte reprezentat în Figura 2.4 a) este $\{(1,8)\}$, până în destinația *9* – $\{(1,2), (2,3), (3,4), (4,7), (7,9)\}$, iar până în destinația *10* – $\{(1,10)\}$. Drumul până în destinația ***8*** din cromozomul-părinte reprezentat în

Figura 2.4 b) este {(1,2), (2,6), (6,7), (7,8)}, până în destinația ***9*** – {(1,2), (2,6), (6,7), (7,9)}, iar până în destinația ***10*** – {(1,2), (2,10)}.



a) b)

Fig. 2.5. Cromozomi-urmași.

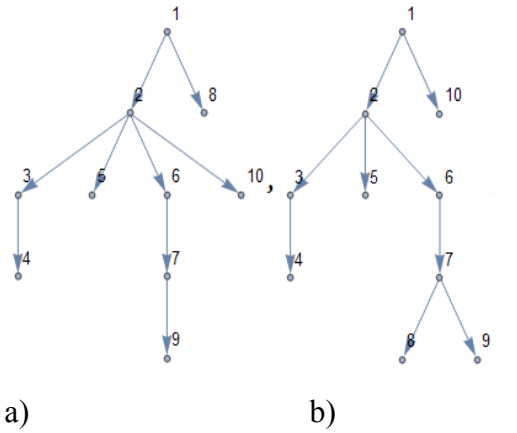
Sursa: elaborat de autor

Pentru a obține cromozomul-urmaș reprezentat în Figura 2.5 a), se unește drumul din sursa 1 până în destinația ***8*** din Figura 2.4 a) cu drumurile din aceeași sursă 1 până în destinațiile ***9*** și ***10*** din Figura 2.4 b). Se adăugă, de asemenea, vârfurile lipsă și drumurile ce le unesc cu sursa 1 în Figura 2.4 a), eliminând repetările posibile ale arcelor. Pentru a obține cromozomul-urmaș reprezentat în Figura 2.5 b), se unește drumul din sursa 1 până în destinația ***8*** din Figura 2.4 b) cu drumurile din sursă până în destinațiile ***9*** și ***10*** din Figura 2.4 a), la care se adaugă vârfurile lipsă cu drumurile ce le unesc cu sursa în Figura 2.4 b), eliminând repetările posibile ale arcelor. Peste graful obținut se construiește un arbore de acoperire. După asemenea operații se obțin doi cromozomi (arbori de acoperire) care pot fi adăugați la populația nouă.

Deci, cromozomii-urmași sunt:

Cromozom-urmaș1 {(1,8), (1,2), (2,6), (6,7), (7,9), (2,10), (2,3), (3,4), (4,5)}

Cromozom-urmaș2 {(1,2), (2,6), (6,7), (7,8), (7,9), (1,10), (2,3), (3,4), (2,5)}



a) b)

Fig. 2.6. Mutația.

Sursa: elaborat de autor

Exemplul (mutație) 2.12. În algoritmul AG3 un rol important îl are mutația. Ea permite generarea de soluții noi și garantează evitarea situațiilor când algoritmul se blochează în minime locale. Se recomandă ca mutația să fie aplicată cu probabilitatea $\varepsilon = 0.5$, în special pentru grafuri de dimensiuni mari. În Figura 2.6 a) și b) sunt reprezentați cromozomi-urmași din Figura 2.5 după mutație. Este aplicată mutația doar asupra cromozomului din Figura 2.6 a), însă cel din Figura 2.6 b) rămâne nemodificat și este adăugat în populația nouă în forma sa originală. În cromozomul-urmaș din Figura 2.6 a) este șters arcul (4,5) și adăugat arcul (2,5) care este prezent în graful inițial.

Deci, dacă este aplicată mutația pentru cromozomul:

Cromozom-urmaș $\{(1,8),(1,2),(2,6),(6,7),(7,9),(2,10),(2,3),(3,4),(4,5)\}$

după mutație se obține cromozomul:

Cromozom-urmaș $\{(1,8),(1,2),(2,6),(6,7),(7,9),(2,10),(2,3),(3,4),(2,5)\}$.

Numărul de iterații necesar pentru stoparea algoritmului depinde în special de dimensiunea rețelei de transport studiată. Pentru obținerea unui cromozom cu caracteristici bune sunt necesare mai multe iterații.

Remarca 2.10. *Algoritmul AG3 se aplică în cazul rețelelor care îndeplinesc următoarele condiții:*

1. *Graful rețelei de transport este conex și aciclic;*
2. *Fiecare vârf al grafului, cu excepția destinațiilor, conține cel puțin un arc ce iese din vârful respectiv;*
3. *Vârfurile destinații nu conțin arce care ies din ele și sunt cel puțin două;*
4. *Există cel puțin un drum din sursă către fiecare dintre destinații.*

Se formulează și se demonstrează următoarele teoreme:

Teorema 2.13. *Algoritmul AG3 necesită un volum de memorie de ordinul* $O(n^2)$.

Demonstrație. *Graful* $G = (V, E)$ *este descris de o listă de adiacență de dimensiunea* $m < n^2$. *Fiecare cromozom constă dintr-o listă* T *de dimensiunea* $n - 1$. *O populație care constă din* $4n$ *cromozomi are dimensiunea* $4n(n-1)$. *Populația este înnoită la fiecare pas. Prin urmare, în momentul construirii unei noi populații acesteia nu-i este alocată memorie suplimentară. Deci. implementarea algoritmului AG3 necesită un volum de memorie de ordinul* $O(n^2)$. □

Teorema 2.14. *Complexitatea unei iterații a algoritmului AG3 este de ordinul* $O(n^2)$.

Demonstrație. *Pentru completarea listei de adiacență care descrie graful* $G = (V, E)$ *sunt necesare* $O(m)$ *operații. Procesul de generare a unui cromozom are complexitatea* $O(n)$. *Generarea întregii populații necesită un număr de operații de ordinul* $O(n^2)$. *Evaluarea soluției asociate unui cromozom necesită* $O(n)$ *operații. Deoarece în populație sunt* $4n$ *cromozomi, complexitatea de evaluare a tuturor cromozomilor este de ordinul* $O(n^2)$. *Încrucișarea și mutația au complexitatea de ordinul* $O(n)$ *pentru un cromozom și complexitatea de ordinul* $O(n^2)$ *pentru întreaga populație. Prin urmare, o iterație a algoritmului AG3 are complexitatea de ordinul* $O(n^2)$. □

Observația 2.3. *Din Teorema 2.14 rezultă că timpul de execuție a algoritmului AG3 este* $O(Un^2)$*, unde* U *este numărul de iterații necesar pentru obținerea unei soluții asociate cromozomului cu caracteristici bune.*

Teorema 2.15. *Algoritmul AG3 converge către un optim local.*

Demonstrație. *Din populația* $P(i-1)$ *sunt transferați în populația* $P(i)$ *cromozomii pentru care valoarea funcției obiectiv în soluția asociată este minimă. Cromozomii populațiilor construite la fiecare iterație reprezintă arbori de acoperire, iar fiecărui cromozom îi este asociată o soluție admisibilă. Se poate spune că după executarea unui număr finit de iterații este depistat cromozomul căruia îi este asociată soluția ce descrie un optim local. Prin urmare, algoritmul AG3 converge către optimul local.* □

2.4.3 Implementarea și testarea algoritmului AG3

Un exemplu practic de aplicare a algoritmului AG3 pentru soluționarea problemei neliniare de transport pe rețea descrisă de o sursă și câteva destinații este prezentat în Anexa 7. Codul scris în Wolfram Mathematica poate fi accesat pe https://github.com/TatianaPasa/GeneticAlgorithm.

Algoritmul AG3 este realizat în limbajul Wolfram și testat pe un șir de exemple – rețele de transport de variate dimensiuni, cu un număr diferit de vârfuri, de destinații și de arce ale grafului care descrie rețeaua.

În Tabelul 2.9 sunt prezentate rezultatele soluționării unui șir de probleme de transport pe o rețea cu n vârfuri, m arce și d destinații, pentru care se cunosc funcțiile concave secvențial-liniare de cost ce depind de fluxul transportat pe fiecare arc și funcția de producere și consum.

Analizând tabelul se constată că $F_T(x)$ întotdeauna descrește de la o iterație la alta, ceea ce presupune că fiecare populație nouă conține mai mulți cromozomi cu caracteristici mai bune decât cei din populația precedentă. Deși se poate observa că la câteva iterații consecutive se repetă aceeași valoare $F(x)$, algoritmul permite ieșirea dintr-un astfel de blocaj determinând un nou cromozom cu caracteristici mai bune, deci și o nouă soluție căreia îi corespunde o valoare mai mică a funcției obiectiv.

Tabelul 2.9. Modificarea valorii $F(x)$ și $F_T(x)$ pentru algoritmul AG3.
Sursa: elaborat de autor

Iterația	**n/m/d** (u. c.)	**n/m/d** (u. c.)	**n/m/d** (u. c.)	**n/m/d** (u. c.)	**n/m/d** (u. c.)	**n/m/d** (u. c.)
	10/23/3	**50/619/10**	**100/2422/20**	**150/4887/50**	**250/14881/70**	**500/57146/100**
I	42/3222	130/36497	317/156257	496/365616	614/774821	841/2066006

II	42/2913	106/32340	246/143659	496/349826	610/733735	786/1951033
III	41/2586	92/29141	231/133008	469/335819	571/698441	780/1853123
IV	39/2417	86/26969	221/123762	425/322657	556/667725	744/1773558
V	26/2110	72/23877	212/116216	425/311097	528/641933	710/1705995
VI	26/1856	72/21886	212/109691	409/299570	528/618762	701/1646399
VII	26/1750	72/20358	205/103967	401/288482	509/598623	679/1597838
VIII	26/1604	72/18900	191/98440	382/277867	499/579515	657/1555438
IX	26/1607	68/16663	180/90601	382/268272	497/562084	655/1516751
X	26/1484	64/15759	175/87347	373/259505	454/545225	653/1482755

În dependență de numărul vârfurilor din graf ce determină dimensiunea unui cromozom se obține un timp de execuție diferit, fapt confirmat și de rezultatele practice prezentate în Tabelul 2.10.

Tabelul 2.10. Timpul de execuție pentru algoritmul AG3. Sursa: elaborat de autor

	m/d	**Timpul** (secunde)		**m/d**	**Timpul** (secunde)
n= 10	20/3	0.2812	**n=100**	2264/15	22.3906
	22/4	0.3125		2527/20	22.6094
	27/5	0.3593		2569/30	22.6719
n=20	101/3	1.0156	**n=150**	4777/30	51.3750
	119/5	1.0312		5178/40	52.1250
	119/7	1.2656		5297/50	52.7813
n=30	203/4	2.1256	**n=200**	9271/50	95.8125
	217/7	2.1406		9291/60	97.8906
	231/9	2.1562		9329/70	96.9375
n=50	590/5	5.4843	**n=300**	20322/70	246.5940
	621/9	5.6093		20663/100	249.2660
	657/12	5.6875		20640/120	250.6720
n=60	860/9	7.9062	**n=400**	37944/80	512.913
	917/12	7.8125		37652/100	489.2340
	972/15	7.8906		38073/120	507.7500
n=70	1191/9	11.0469	**n=500**	59771/100	918.531
	1206/15	11.2344		60756/120	917.5630

	1312/20	11.2500		60012/150	918.4380

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

Fiecare test presupune execuția a 10 iterații ale algoritmului pe grafuri cu n vârfuri, m arce și d destinații. Este confirmată experimental Teorema 2.14, deoarece se poate observa o legătură directă între timpul de execuție a algoritmului și numărul de vârfuri. Numărul de arce și numărul de destinații nu influențează timpul de execuție.

2.5 Soluționarea problemei neliniare de transport pe rețea cu câteva surse și câteva destinații utilizând algoritmi genetici

Este cunoscută situația când mai multe fabrici sau uzine care produc produsul X deservesc câțiva cumpărători la care produsul ajunge printr-un șir de puncte intermediare astfel încât costul de transport să fie minim. Această structură poate fi descrisă de o rețea de transport cu câteva surse, câteva destinații și o mulțime de puncte intermediare, pentru care se cere minimizarea costului de transport. În surse produsul se produce, în destinații el se consumă, iar în punctele intermediare nu se produce și nici nu se consumă, dar respectă condiția de conservare a fluxului, ceea ce înseamnă că în punctul intermediar cantitatea de produs care intră este egală cu cantitatea de produs care iese din el.

O modalitate de soluționare a problemei de transport cu cost minim constă în introducerea unei surse fictive (super-sursă) care este conectată cu toate sursele prin arce, căror le sunt asociate funcții de cost nule, și în introducerea unei destinații fictive (super-destinație) care este conectată cu toate destinațiile prin arce, căror le sunt asociate funcții de cost nule. Evident, o asemenea modificare a rețelei de transport conduce la necesitatea apariției unor restricții suplimentare în modelul matematic al problemei care să asigure producerea în fiecare surse și consumul în fiecare destinație. În lipsa unor asemenea restricții, soluția obținută nu neapărat permite distribuirea produsului în toate destinațiile cu necesarul de produs transportat din super-sursa rețelei, deoarece fluxul de produs poate să nu treacă prin toate sursele rețelei inițiale.

După cum s-a menționat anterior, algoritmii propuși și analizați în secțiunea 2.2 (Lozovanu & Pasha, 2002), (Pașa & Ungureanu, 2017a), (Pașa & Ungureanu, 2017d), (Pașa & Ungureanu, 2017e), (Pașa & Ungureanu, 2017f), deși permit soluționarea problemei într-un timp de execuție mai mic decât în cazul funcțiilor standard din Sistemul Mathematica (Pașa, 2018a), pentru rețele de dimensiuni mari solicită timp de execuție nepermis de mare.

2.5.1 Formularea problemei

În cele ce urmează se consideră problema neliniară de transport (2.1) – (2.3) pe rețea cu câteva surse și câteva destinații. Rețeaua de transport este descrisă de mulțimea de vârfuri $V = V_s \cup V_t \cup V_{int}$, unde $|V| = n + m + g$ și mulțimea de arce $E \subset V \times V$, $|E| = u$, unde V_s este mulțimea de surse, $|V_s| = n$, V_t – mulțimea de destinații, $|V_t| = m$ iar V_{int} – mulțimea de puncte intermediare, $|V_{int}| = g$.

Mulțimii de vârfuri i se asociază funcția de producere și consum de forma:

$$q(v) = 0 \text{ pentru } v \epsilon V \setminus (V_s \cup V_t)$$
$$\sum_{i=1}^{n} q(v_i) = \sum_{j=1}^{m} q(v_j) \qquad (2.10)$$

unde $v_i \in V_s, i = \overline{1,n}$, este sursa i a rețelei de transport, $v_j \in V_t, j = \overline{1,m}$, – destinația j a rețelei de transport.

Se cunoaște din 2.1 că problema neliniară de transport pe rețea (2.1) – (2.3), (2.10) cu funcții concave secvențial-liniare de cost poate fi soluționată utilizând algoritmi finiți ce se bazează pe cercetarea tuturor grafurilor fără cicluri. În cazul problemei descrise de grafuri dense și/sau de dimensiuni mari o astfel de abordare este imposibilă din cauza insuficienței de timp sau a echipamentului de calcul care nu permite prelucrarea unor date atât de voluminoase. Se recomandă utilizarea algoritmilor genetici care permit păstrarea candidaților promițători, eliminarea pretendenților cu costuri mari, astfel îngustând semnificativ mulțimea soluțiilor admisibile care sunt verificate la optimalitate.

2.5.2 Algoritmul genetic AG4

Deoarece aplicarea operatorilor de încrucișare și mutație poate conduce la situații în care nu sunt respectate restricțiile de existență a fluxului în rețea, sunt utilizate metode din teoria grafurilor care permit aplicarea corectă a acestor operatori.

Deoarece dimensiunea populației trebuie să permită exploatarea și explorarea soluțiilor admisibile astfel încât să fie obținut un rezultat bun în timp rezonabil, numărul de cromozomi din populație este $4nm$. Fiecare cromozom constă din 3 compartimente: *primul* conține n arbori de acoperire fiecare cu rădăcina în una dintre sursele rețelei; *al doilea* este o permutare a numerelor $i = \overline{1,n}$ asociate surselor din rețea și descrie ordinea lor de deservire; *al treilea* este o permutare a numerelor $j = \overline{1,m}$ asociate destinațiilor din rețea și descrie ordinea lor de deservire. Cromozomii descriși prin tripletele menționate permit obținerea după decodificarea unor soluții

admisibile ale problemei de optimizare neliniară pe rețea de transport cu câteva surse și câteva destinații.

Algoritmul genetic AG4 propus pentru soluționarea problemei de transport cu funcții concave nedescrescătoare secvențial-liniare de cost constă din următorii pași:

Pasul 1 *Inițializare.* Se generează aleatoriu populația inițială din $4nm$ cromozomi: fiecare cromozom este descris de o listă de forma: $P = \left\{\{T_r, r = \overline{1,n}\}, P_n, P_m\right\}$, unde $T_r = \{(i,j) \in E\}$, $r = \overline{1,n}$ - arbori de acoperire cu rădăcina în sursa r, P_n=$\{p(1), p(2), \dots, p(n)\}$ - ordinea de deservire a surselor, $P_m = \{p(1), p(2), \dots, p(m)\}$ - ordinea de deservire a destinațiilor.

Pasul 2 *Decodificarea și Evaluarea cromozomilor.* Decodificarea presupune asocierea fiecărui cromozom cu o soluție admisibilă având la bază modul de codificare a problemei.

Se inițializează tablourile unidimensionale care conțin respectiv cantitățile de produs în surse $\alpha' = [\alpha'_1, \alpha'_2, \dots, \alpha'_n]$ și necesitățile de produs în destinații $\beta' = [\beta'_1, \beta'_2, \dots, \beta'_m]$.

Pentru fiecare element $r \in P_n$ se construiește soluția intermediară pe arborele de acoperire corespunzător:

1. Pentru fiecare $j \in P_m$ avem:

$$x_{rh} = min\{\alpha'_r, \beta'_j\},$$

$v_j \in V_t$ și h - indicele arcului $(i,j) \in T_r$. Apoi, se actualizează $\alpha'_r := \alpha'_r - x_{rh}$, $\beta'_j := \beta'_j - x_{rh}$.

2. Arcului (i,j) cu indicele h unde $j \notin V_t$ i se asociază fluxul:

$$x_{rh} = \begin{cases} 0, & (i,j) \notin T_r, \\ \sum\limits_{(j,j') \in T_r} x_{rh'} & h' - indicele\ arcului\ (j,j'). \end{cases}$$

Ca rezultat se obțin soluțiile intermediare de forma:

$$x_{rh} = (x_{r1}, x_{r2}, \dots, x_{ru}), r = \overline{1,n}$$

iar soluția finală a problemei are forma:

$$x = \left(\sum_{i=1}^{n} x_{i1}, \sum_{i=1}^{n} x_{i2}, \dots, \sum_{i=1}^{n} x_{iu}\right).$$

Evaluarea cromozomilor presupune determinarea valorii funcției obiectiv pentru fiecare dintre soluțiile obținute asociate cromozomilor.

Pasul 3 *Selectarea cromozomilor.* Cromozomii sunt sortați în ordinea creșterii valorii funcției obiectiv pentru soluția asociată cromozomului. Cromozomii din prima jumătate a populației sunt transferați în noua populație.

Pasul 4 *Încrucișarea cromozomilor.* Încrucișarea are loc între cromozomii transferați în populația $P(i)$ din populația $P(i-1)$. Aleatoriu este aplicată aceeași tăietură ambilor cromozomi-părinți asupra compartimentului unu.

Încrucișarea cromozomilor presupune:

- primul cromozom-urmaș este descris de arborii de acoperire de până la tăietură a cromozomului-mamă cu ordinea respectivă de deservire a surselor și arborii de acoperire de după tăietură a cromozomului-tată cu ordinea respectivă de deservire a destinațiilor;
- al doilea cromozom-urmaș este descris de arborii de acoperire de până la tăietură a cromozomului-tată cu ordinea respectivă de deservire a surselor și arborii de acoperire de după tăietură a cromozomului-mamă cu ordinea respectivă de deservire a destinațiilor.

Fiecare pereche de cromozomi-părinți generează doi cromozomi-urmași și dimensiunea populației rămâne constantă.

Pasul 5 *Mutația.* Unei gene a cromozomului-urmaș i se aplică o mutație cu probabilitatea $\varepsilon \in [0.1, 0.5]$ și presupune:

- pentru fiecare dintre compartimentele *doi,* descris de ordinea de deservire a surselor, și *trei,* descris de ordinea de deservire a destinațiilor, aleatoriu se decide dacă este aplicată mutația;
- în fiecare dintre compartimentele alese se schimbă valorile între două elemente alese aleatoriu.

Celelalte gene rămân nemodificate.

Pasul 6 *Verificarea condiției de oprire.* Oprirea algoritmului are loc după k iterații. În calitate de soluție a problemei servește soluția ce corespunde cromozomului pentru care valoarea funcției obiectiv din ultima populație este minimă. Se trece la *Pasul 2* dacă condiția de oprire nu este satisfăcută.

Remarca 2.11. *În cazul problemelor de dimensiuni mari, în calitate de condiție de oprire a execuției algoritmului poate servi durata de execuție în timp. După o perioadă anumită de timp este stopată generarea noilor populații, iar ca soluție a problemei servește soluția ce corespunde cromozomului cu cele mai bune caracteristici din ultima populație.*

Remarca 2.12. *Algoritmul AG4 se aplică în cazul rețelelor care îndeplinesc următoarele condiții:*

1. *Graful rețelei de transport este conex și aciclic;*

2. *Există cel puțin două surse și două destinații;*
3. *Există cel puțin un punct intermediar;*
4. *Există cel puțin un drum din fiecare sursă în fiecare destinație;*
5. *Vârfurile surse nu conțin arce ascendente;*
6. *Vârfurile destinații nu conțin arce descendente.*

Cunoscându-se cantitatea de flux ce intră în fiecare dintre destinații, se aplică procedura de căutare în adâncime pentru a determina soluția asociată cromozomului și valoarea funcției obiectiv în această soluție. Fiecare arbore de acoperire este parcurs repartizând fluxul prin arce. Când se întâlnesc în același vârf două (sau mai multe) arce care au punctul final ce aparține drumurilor în diferite destinații, arcului care intră în acel vârf îi este asociat fluxul total care trebuie să treacă prin arcele respective. Astfel, fluxul total care trebuie transportat în destinații coincide cu mărimea fluxului ce pornește din surse.

Exemplul (codificare) 2.13. Se consideră rețeaua de transport, din Figura 2.7, descrisă de graful cu mulțimea de vârfuri $\{1,2,3,4,5,6\}$, unde $\{1,2\}$ - mulțimea de surse iar $\{5,6\}$ - mulțimea de destinații, și mulțimea de arce $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$. Se cere de transportat fluxul de produs de 100 u. c. din surse, cu capacitățile respectiv de 30 u. c. pentru sursa 1 și 70 u. c. pentru sursa 2, în destinațiile cu necesitățile de 60 u. c. pentru destinația 5 și 40 u. c. pentru destinația 6, astfel încât costul de transport să fie minim.

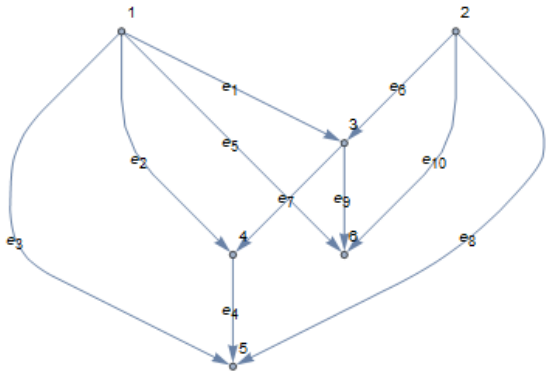


Fig. 2.7. Rețea cu două surse și două destinații.
Sursa: elaborat de autor

Un cromozom a populației, în acest caz este descris de:

- *compartimentul 1* care conține doi arbori de acoperire $T_1 = \{(1,3),(1,4),(4,5),(1,6)\}$ și $T_2 = \{(2,3),(3,4),(2,5),(3,6)\}$ care au rădăcina în sursa 1 și 2 respectiv;
- *compartimentul 2* care conține ordinea de deservire a surselor, fie $\{1,2\}$.
- *compartimentul 3* care conține ordinea de deservire a destinațiilor, fie $\{2,1\}$ ceea ce înseamnă că este aprovizionată destinația 6 (a doua) și apoi destinația 5 (prima).

Astfel, se obține un cromozom de forma:

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

Exemplul (decodificare) 2.14. Pentru cromozomul generat mai sus, trebuie de asociat o soluție admisibilă ceea ce înseamnă că se cere repartizarea fluxului de produs prin fiecare arc încât din fiecare sursă să iasă tot fluxul și să ajungă în destinații astfel încât să fie aprovizionate ambele destinații. Inițial, pentru fiecare arbore asociem indicii care arată la al câtelea arc ne referim ca ulterior să plasăm cantitatea de flux ce trebuie să treacă prin acel arc astfel: $T_1 = \{(1,3),(1,4),(4,5),(1,6)\} = \{e_1, e_2, e_4, e_5\}$ și $T_2 = \{(2,3),(3,4),(2,5),(3,6)\} = \{e_6, e_7, e_8, e_9\}$. Soluția admisibilă $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10})$ se obține astfel:

Se inițializează tablourile: $\alpha' = [30, 70]$, $\beta' = [60, 40]$.

Pentru $r = 1$: 1) $x_{51} = min\{30,40\} = 30$, $\alpha'_1 := 30 - 30 = 0$, $\beta'_2 := 40 - 30 = 10$;

$x_{41} = min\{0,60\} = 0$, $\alpha'_1 := 0 - 0 = 0$, $\beta'_1 := 60 - 0 = 60$.

2) $x_{11} = 0$, $x_{21} = 0$, $x_{31} = 0$, $x_{61} = 0$, $x_{71} = 0$, $x_{81} = 0$, $x_{91} = 0$, $x_{10\,1} = 0$.

Pentru $r = 2$: 1) $x_{91} = min\{70,10\} = 10$, $\alpha'_2 := 70 - 10 = 60$, $\beta'_2 := 10 - 10 = 0$;

$x_{81} = min\{60,60\} = 60$, $\alpha'_2 := 60 - 60 = 0$, $\beta'_1 := 60 - 60 = 0$.

2) $x_{12} = 0$, $x_{22} = 0$, $x_{32} = 0$, $x_{62} = x_{72} + x_{92} = 0 + 10 = 10$, $x_{52} = 0$, $x_{72} = 0$, $x_{62} = x_{72} + x_{92} = 0 + 10 = 10$, $x_{10\,1} = 0$.

Deci, $x_1 = x_{11} + x_{12} = 0$, $x_2 = x_{21} + x_{22} = 0$, $x_3 = x_{31} + x_{32} = 0$, $x_4 = x_{41} + x_{42} = 0$, $x_5 = x_{51} + x_{52} = 30 + 0 = 30$, $x_6 = x_{61} + x_{62} = 0 + 10 = 10$, $x_7 = x_{71} + x_{72} = 0$, $x_8 = x_{81} + x_{82} = 0 + 60 = 60$, $x_9 = x_{91} + x_{92} = 0 + 10 = 10$, $x_{10} = x_{10\,1} + x_{10\,2} = 0$.

Soluția admisibilă obținută este: $x = (0,0,0,0,30,100,0,60,10,0)$.

Cunoscându-se cantitatea de flux ce trece prin fiecare dintre arcele rețelei, descris de soluția admisibilă, se calculează costul de transport pe fiecare arc, se însumează și se obține valoarea funcției obiectiv, care și este costul total de transport pe rețea.

Exemplul (încrucișare) 2.15. Pentru rețeaua din Figura 2.7, se consideră doi cromozomi-părinți care participă la încrucișare. Tăietura este aplicată la mijlocul compartimentului unu:

Cromozom-mama {{1→3,1→4,4→5,1→6},{2→3,3→4,2→5,3→6}},{1,2},{2,1}}

Cromozom-tata {{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1}}.

Astfel, se obțin următorii cromozomi-urmași:

Cromozom-urmaș1 {{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1}}

Cromozom-urmaș2 {{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,3→6}},{1,2},{2,1}}.

Exemplul (mutație) 2.16. În acest exemplu se alege compartimentul destinațiilor, asupra căruia se aplică mutația:

Cromozom-urmaș {{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1}}

pentru care sunt schimbate cu locul două elemente și se obține:

Cromozom-urmaș {{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{1,2}.

Revenind la cadrul general, se formulează și se demonstrează următoarele teoreme:

Teorema 2.16. *Algoritmul AG4 necesită un volum de memorie de ordinul* $O(|V|(n|V| + n + m))$.

Demonstrație. *Datele de intrare descrise de tabelul de restricții au dimensiunea* u. *Pentru păstrarea soluției asociate cromozomului cu cele mai bune caracteristici din populație este necesară memorie de volum* umn. *Păstrarea unui cromozom necesită* $n|V| + n + m$ *unități de memorie, iar pentru întreaga populație se cer* $4|V|(n|V| + n + m)$ *unități de memorie. Prin urmare, algoritmul AG4 necesită un volum de memorie de ordinul* $O(|V|(n|V| + n + m))$.□

Teorema 2.17. *Complexitatea unei iterații a algoritmului AG4 este de ordinul* $O(n|V|(m + |V| + u))$ *operații.*

Demonstrație. *Pentru inițializarea datelor de intrare descrise de tabelul de restricții sunt necesare* $O(u)$ *operații. Generarea unui cromozom necesită* $O(n|V| + n + m)$ *operații, iar generarea întregii populații necesită* $O(|V|(n|V| + n + m))$ *operații. Evaluarea soluției asociate unui cromozom necesită* $O(n(m + |V| + u))$ *operații. Deoarece în populație sunt* $4|V|$ *cromozomi, complexitatea de evaluare a tuturor soluțiilor asociate cromozomilor este* $O(n|V|(m + |V| + u))$. *Încrucișarea are o complexitate* $O(n|V| + n + m)$ *pentru un cromozom, iar pentru întreaga populație –* $O(|V|(n|V| + n + m))$. *Prin urmare, o iterație a algoritmului AG4 are complexitatea* $O(n|V| + n + m)$. □

Observația 2.4. *Din Teorema 2.17 rezultă că timpul de execuție a algoritmului AG4 este* $O(U(n|V| + n + m))$, *unde* U *este numărul de iterații necesar pentru obținerea unei soluții asociate cromozomului cu caracteristici bune.*

Teorema 2.18. *Algoritmul AG4 converge către un optim local.*

Demonstrație. *Din populația* $P(i-1)$ *sunt transferați în populația* $P(i)$ *cromozomii pentru care valoarea funcției obiectiv în soluția asociată este minimă. Cromozomilor populațiilor construite la fiecare iterație li se asociază câte o soluție admisibilă. Deci, se poate spune că după executarea unui număr finit de iterații este depistat cromozomul căruia îi este asociată soluția ce descrie optimul local. Prin urmare, algoritmul AG4 converge către un optim local.* □

2.5.3 Implementarea și testarea algoritmului AG4

Un exemplu practic de aplicare a algoritmului AG4 pentru soluționarea problemei neliniare de transport pe rețea descrisă de câteva surse și câteva destinați este prezentat în Anexa 8. Pe https://github.com/TatianaPasa/GeneticAlgorithm poate fi accesat codul scris în Wolfram Mathematica.

Algoritmul AG4 este realizat în limbajul Wolfram și testat în baza unui șir de exemple – rețele de transport de variate dimensiuni, cu un număr de surse n și număr de destinații m ale rețelei de transport. În Tabelele 2.11 și 2.12 sunt prezentate rezultatele soluționării unui șir de probleme de diferite dimensiuni, unde funcțiile de cost sunt funcții concave secvențial-liniare care depind de fluxul transportat pe fiecare arc.

Pentru fiecare din problemele formulate sunt generate 10 populații. Se observă cum se modifică valoarea funcției obiectiv calculată pentru soluția asociată cromozomului cu cele mai bune caracteristici și valoarea funcției obiectiv totale a populației pentru populația generată la fiecare iterație. Micșorarea acestor valori de la o iterație la alta denotă o îmbunătățire a soluției obținute deci și o convergență a algoritmului către o soluție pseudo optimă.

Tabelul 2.11. Modificarea valorii $F(x)$ și $F_T(x)$ pentru algoritmul AG4. Sursa: elaborat de autor

Iterația	**n / m** (u. c.)					
	3 / 3	**7 / 7**	**10 / 10**	**20 / 20**	**30 / 30**	**50 / 50**
I	15 / 958	27 / 2576	46 / 6047	157 / 53298	244 / 179817	302 / 280233
II	15 / 868	27 / 2508	46 / 5852	157 / 52242	244 / 177927	302 / 277529
III	13 / 785	27 / 2422	46 / 5632	157 / 51861	244 / 175973	302 / 275198
IV	13 / 695	26 / 2383	41 / 5448	157 / 51452	241 / 174691	302 / 273723
V	13 / 696	23 / 2278	41 / 5280	146 / 50972	241 / 173351	302 / 272486
VI	11 / 666	23 / 2247	37 / 5162	145 / 50451	241 / 172435	298 / 271536
VII	11 / 677	23 / 2149	36 / 5011	145 / 50239	235 / 17673	292 / 270169
VIII	10 / 627	23 / 2020	36 / 4938	143 / 49773	235 / 170716	292 / 269421
IX	10 / 594	23 / 2013	36 / 4904	143 / 49249	235 / 168986	292 / 268906
X	10 / 589	23 / 2015	36 / 4775	143 / 48661	235 / 168618	287 / 267521
Necunoscute	38	94	210	2035	10305	17450

Din Tabelul 2.11 se observă că $F_T(x)$ descrește de la o iterație la alta, cu unele excepții când în urma încrucișării și mutației se obțin mai multe soluții cărora le corespund costuri mai ridicate. Mărirea nesemnificativă a $F_T(x)$ este urmată de descreșteri, fapt care sugerează că la pașii

următori se obțin rezultate mai bune în populațiile noi. Deși în câteva iterații consecutive se observă că se repetă aceeași valoare $F(x)$ pentru aceeași soluție, se atrage atenția că algoritmul permite ieșirea dintr-un astfel de blocaj determinând prin mutații un nou cromozom cu caracteristici mai bune.

În Tabelul 2.12 este dat timpul de execuție a algoritmului AG4 pentru probleme de diferite dimensiuni, pentru a căror soluționare sunt utilizate de la 38 de necunoscute pentru problema cu 3 surse și 3 destinații până la 17450 de necunoscute pentru problema cu 50 de surse și 50 de destinații.

Tabelul 2.12. Timpul de execuție pentru algoritmul AG4. Sursa: elaborat de autor

n / m (secunde)	**3 / 3**	**5 / 5**	**5 / 7**	**5 / 9**	**7 / 7**	**7 / 9**
	0.3906	0.7500	1.0625	1.6093	1.8906	2.8437
	0.4062	0.8593	1.0312	1.5156	1.9062	2.7187
	0.4531	0.7656	1.0000	1.5625	1.8593	2.7968
	0.3593	0.8125	1.0468	1.6562	1.8750	2.7187
	0.4218	0.6875	0.9687	1.5937	2.0000	2.7500
n / m (secunde)	**10 / 10**	**10 / 15**	**10 / 20**	**20 / 20**	**30 / 30**	**50 / 50**
	5.2812	11.5625	26.9219	90.8594	501.8130	1660.4200
	5.2031	11.7031	28.2969	99.9688	515.1880	1562.6400
	5.3125	12.0313	27.5156	93.8594	515.6560	1556.1100
	5.4375	12.2813	27.5625	97.5156	495.8440	1562.1100
	5.4062	11.9688	28.5625	93.3594	494.3750	1555.3300

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

Se observă că algoritmul permite obținerea unor soluții locale în timp rezonabil în cazul rețelelor de transport de dimensiuni mari. În Tabelul 2.12 este reflectată o creștere a timpului de execuție direct proporțională cu numărul de surse și destinații.

2.6 Concluzii la capitolul 2

Capitolul 2 conține un studiu a problemelor neliniare de transport pe rețea cu o sursă și o destinație, pe rețea cu o sursă și câteva destinații, pe rețea cu câteva surse și câteva destinații.

Pentru problemele formulate sunt propuși algoritmi de soluționare care permit obținerea soluției în timp rezonabil chiar și în cazul problemelor de dimensiuni mari.

Rezultatele obținute în capitolul 2 completează rezultatele cunoscute la moment în domeniul soluționării problemei neliniare de transport pe rețea cu funcții concave de cost și se referă la:

1) elaborarea algoritmilor AE1 și AE2 de soluționare a problemei neliniare de transport pe rețea cu o sursă și o destinație, pe rețea cu o sursă și câteva destinații și pe rețea cu câteva surse și câteva destinații, descrise de funcții concave de cost care presupun reducerea problemei neliniare la o problemă liniară;
2) elaborarea algoritmilor AG1, AG2, AG3 și AG4 de soluționare a problemei neliniare de transport pe rețea cu o sursă și cu o destinație, pe rețea cu o sursă și câteva destinații și pe rețea cu câteva surse și câteva destinații, descrise de funcții concave de cost care au la bază aplicarea operatorilor de selectare, încrucișare și mutație a algoritmilor genetici;
3) descrierea unei codificări corecte a problemei respective, pentru fiecare dintre algoritmii genetici AG1, AG2, AG3 și AG4 de mai sus, astfel încât la decodificare se obține o singură soluție admisibilă pentru fiecare cromozom al populației;
4) descrierea operatorilor de încrucișare și mutație, astfel încât ori de câte ori sunt aplicați acești operatori, în cadrul fiecărui dintre algoritmii genetici AG1, AG2, AG3 și AG4, după decodificarea cromozomului nou construit se obține o soluție admisibilă a problemei codificate;
5) demonstrarea convergenței algoritmilor AE1, AE2, AG1, AG2, AG3 și AG4 către un optim local;
6) determinarea volumului necesar de memorie pentru implementarea practică a algoritmilor elaborați;
7) testarea algoritmilor implementați în Sistemul Mathematica care permit soluționarea problemelor neliniare de transport.

Potrivit studiului efectuat în acest capitol, se formulează următoarele concluzii:

1. Algoritmul AE1 permite soluționarea problemei neliniare de transport pe rețea cu funcții concave de cost, prin reducerea acesteia la o problemă liniară, în timp rezonabil pentru probleme cu zeci de necunoscute.
2. Algoritmul AE2 permite soluționarea problemei neliniare de transport pe rețea cu funcții concave de cost prin reducerea acesteia la o problemă liniară în timp rezonabil și aplicarea

calculului secvențial sau paralel care permite îmbunătățirea soluției în comparație cu rezultatele furnizate de algoritmul AE1 pentru soluționarea aceleiași probleme.

3. Soluția problemei neliniare de transport pe rețea, obținută în urma aplicării algoritmilor AE1 și AE2, depinde de soluția admisibilă inițială obținută la *Pasul 1*, fapt care presupune că o alegere corectă a soluției inițiale duce la obținerea unei soluții mai bune.
4. Timpul de execuție, în cazul utilizării calculului paralel pentru algoritmul AE2, nu se îmbunătățește proporțional cu numărul de procesoare utilizate, în comparație cu aplicarea calculului secvențial.
5. Algoritmii genetici AG1, AG2, AG3 și AG4 propuși pentru soluționarea problemelor neliniare de transport formulate reduc esențial timpul de obținere a soluției, astfel încât pot fi rezolvate probleme cu câteva mii de necunoscute.
6. Complexitatea unei iterații pentru fiecare dintre algoritmii genetici AG1, AG2, AG3 și AG4 este una polinomială.
7. Fiecare dintre algoritmii AG1, AG2, AG3 și AG4 generează o soluție pseudo optimă care este o soluție locală din vecinătatea soluției optime globale în cazul generării unui număr suficient de populații.
8. Pentru algoritmii AG1, AG2, AG3 și AG4 este demonstrat experimental că permit ieșirea din blocaje în soluții locale astfel încât după câteva iterații este depistat un cromozom cu caracteristici mai bune căruia îi este asociată o soluție pentru care costul de transport este mai mic. Această îmbunătățire se datorează operațiilor de selecție, încrucișare și mutație descrise pentru fiecare algoritm.
9. Soluția obținută ca urmare a implementării algoritmilor AG1, AG2, AG3 și AG4 depinde de populațiile generate la *Pasul 1*.
10. Pentru algoritmul AG3 experimental este confirmată legătura directă dintre complexitatea unei iterații și numărul de vârfuri a rețelei.
11. Algoritmii descriși în acest capitol permit soluționarea problemelor respective în timp rezonabil chiar și pentru probleme de dimensiuni mari.

III. PROBLEMA NELINIARĂ DE TRANSPORT CU 4 ȘI 5 INDICI

În capitolul precedent sunt expuse rezultatele ce se referă la algoritmi de soluționare a problemei neliniare de transport pe rețea cu funcții concave de cost prin care este transportat un flux omogen de produs nefiind cunoscut ce transport este utilizat.

Managementul aprovizionării devine tot mai important în economie pentru funcționarea eficientă a companiilor. Problemele care presupun cost minim de transport a produselor pot varia în dependență de tipurile de produse (materiale de construcții, produse alimentare, produse petroliere) transportate, de numărul și tipul transportului utilizat pentru transportare (Diaz-Para, Ruiz-Vanoye, Loranca, Fluentes-Penna, & Barrera-Camara, 2014) (tren, vapor, camion, avion), de numărul de surse și destinații, dar și de calea de transport (aer, apă, drumuri, țevi, cabluri) aleasă de transportator.

În acest capitol este formulată și soluționată problema neliniară de transport pe rețea cu funcții concave de cost cu 4 indici descrisă de surse, destinații, tipuri de produse și tipuri de transport care circulă prin rețeaua de transport studiată. Tot în acest segment de investigație este formulată și soluționată problema de transport pe rețea cu funcții concave de cost cu 5 indici, descrisă de surse, destinații, puncte intermediare, tipuri de produse și tipuri de transport, care este formulată și soluționată ca problemă cu 3 indici descrisă de arce, tipuri de produse și tipuri de transport. Pentru ambele cazuri se cere minimizarea costurilor de transport. Rezultatele prezentate în acest cadru al demersului nostru științific publicate în lucrările (Pașa, 2018b), (Pașa & Ungureanu, 2019a), (Pașa, 2019b), (Pașa, 2020b) împreună cu cele anunțate în capitolul doi acoperă toate problemele enunțate în primul capitol.

3.1 Problema de transport cu mai mulți indici

3.1.1 Formularea problemei. Proprietăți

În cele ce urmează, este considerată problema de transport pe rețea cu funcții concave de cost descrisă de:

- n surse $A_1, A_2, \dots, A_n$ care posedă cantitățile de produs respectiv $\alpha_1, \alpha_2, \dots, \alpha_n$;
- m destinații $B_1, B_2, \dots, B_m$ cu necesitățile respective de produs $\beta_1, \beta_2, \dots, \beta_m$;
- p tipuri de produse $P_1, P_2, \dots, P_p$ cu cantitățile respective $\gamma_1, \gamma_2, \dots, \gamma_p$;
- q tipuri de transport $T_1, T_2, \dots, T_q$ cu capacitățile de transport respective $\delta_1, \delta_2, \dots, \delta_q$.

Costul de transport este descris de funcții neliniare $\varphi_{ijkl}(x_{ijkl})$ care depind de cantitatea de produs transportat de la surse la destinații. Fluxul x_{ijkl} descrie cantitatea de produs P_l transportat din sursa A_i în destinația B_j utilizând tipul de transport T_k. Scopul problemei este de a minimiza costul total de transport al tuturor produselor disponibile în surse pentru aprovizionarea tuturor destinațiilor. Această problemă face parte din grupul de probleme de transport cu 4 indici (PT4I), unde în calitate de cei 4 indici sunt: sursele, destinațiile, tipurile de transport și tipurile de produse.

Problema neliniară de transport cu 4 indici (PNT4I) constă în determinarea unui flux x^* ce minimizează funcția:

$$F(x) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p}\sum_{l=1}^{q}\varphi_{ijkl}\left(x_{ijkl}\right),$$

unde $\varphi_{ijkl}(x_{ijkl})$ sunt funcții concave secvențial-liniare nedescrescătoare de cost, adică se cere soluționarea problemei neliniare:

$$F(x) \to min, \tag{3.1}$$

$$\begin{cases} \sum_{j=1}^{m}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ijkl} = \alpha_i, \quad i = \overline{1,n}, \\ \sum_{i=1}^{n}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ijkl} = \beta_j, \quad j = \overline{1,m}, \\ \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{q} x_{ijkl} = \gamma_k, \quad k = \overline{1,p}, \\ \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p} x_{ijkl} = \delta_l, \quad l = \overline{1,q}, \\ x_{ijkl} \geq 0, \quad (i,j,k,l), \end{cases} \tag{3.2}$$

pentru care sunt respectate condițiile de pozitivitate; deci, mărimile $\alpha_1,\ \alpha_2, \dots, \alpha_n, \beta_1,\ \beta_2, \dots, \beta_m$, $\gamma_1,\ \gamma_2, \dots, \gamma_p$ și $\delta_1,\ \delta_2, \dots, \delta_q$ sunt pozitive și nenule.

Forma generală a soluției optime pentru PNT4I descrisă de (3.1) – (3.2) este $x^* = (x^*_{1111}, x^*_{1112}, \dots, x^*_{nmpq})$. Se presupune că toate datele sunt valori reale și se dorește să se obțină ca soluție optimă o valoare reală fezabilă.

Deoarece funcția de cost a problemei formulate este o funcție separabilă și neliniară, iar restricțiile sunt funcții separabile liniare, problema este neliniară separabilă și poate fi soluționată utilizând metodele respective.

Definiția 3.1. Spunem că problema de transport PNT4I este echilibrată și admite soluții dacă satisface condițiile:

$$\sum_{i=1}^{n} \alpha_i = \sum_{j=1}^{m} \beta_j = \sum_{k=1}^{p} \gamma_k = \sum_{l=1}^{q} \delta_l = H, \tag{3.3}$$

$$\alpha_i > 0, \beta_j > 0, \gamma_k > 0, \delta_l > 0.$$

În cazul în care una dintre egalități nu se respectă, se adaugă un punct fictiv pentru a compensa restricția de egalitate, iar costul de transport este considerat zero. Se determină mărimea $H = max\{\sum_{i=1}^{n} \alpha_i, \sum_{j=1}^{m} \beta_j, \sum_{k=1}^{p} \gamma_k, \sum_{l=1}^{q} \delta_l\}$ și, în dependență de situație, poate fi:

1. Dacă $\sum_{i=1}^{n} \alpha_i < H$, se adaugă un nod $n+1$ cu $\alpha_{i+1} = H - \sum_{i=1}^{n} \alpha_i$. Deoarece acest nod $n+1$ lipsește, nu este transportat nici un produs din această sursă și costul de transport este descris de funcția $\varphi_{(n+1)jkl}(x) = 0$;
2. Dacă $\sum_{j=1}^{m} \beta_j < H$, se adaugă un nod $m+1$ cu $\beta_{j+1} = H - \sum_{j=1}^{m} \beta_j$. Deoarece acest nod $m+1$ lipsește, nu este transportat un produs în această destinație și costul de transport este descris de funcția $\varphi_{i(m+1)kl}(x) = 0$;
3. Dacă $\sum_{k=1}^{p} \gamma_k < H$, se adaugă un produs $p+1$ cu $\gamma_{p+1} = H - \sum_{k=1}^{p} \gamma_k$. Deoarece acest produs lipsește, atunci el nu este transportat și costul de transport este descris de funcția $\varphi_{ij(p+1)l}(x) = 0$;
4. Dacă $\sum_{l=1}^{q} \delta_l < H$, se adaugă un tip de transport $q+1$ cu $\delta_{q+1} = H - \sum_{l=1}^{q} \delta_l$. Deoarece acest tip de transport lipsește, nu poate fi utilizat pentru a transporta produse și costul de transport este descris de funcția $\varphi_{ijk(q+1)}(x) = 0$.

După modificările efectuate, este soluționată problema neliniară de transport ce conține nodul, produsul sau tipul de transport fictiv adăugat, iar sistemul de soluții admisibile (3.2) conține și acel element nou. Dacă în soluția optimă a problemei modificată apare o mărime nenulă asociată sursei $n+1$, destinației $m+1$, tipului de produs $p+1$ sau tipului de transport $q+1$ atunci când este descrisă soluția optimă pentru problema inițială, acea mărime se înlocuiește cu zero.

De exemplu, dacă pentru produsul $p+1$ sunt $x_{23(p+1)4} = 5$ unități de produs, acestea se înlocuiesc cu $x_{23(p+1)4} = 0$. Acest fapt este condiționat de situația că nu pot fi transportate 5 unități de produs din sursa 2 în destinația 3 cu tipul de transport 4, deoarece acest produs $p+1$ lipsește.

În cazul în care costul de transport al unui produs P_k, $k = \overline{1,p}$ cu ajutorul tipului de transport T_l, $l = \overline{1,q}$ din sursa A_i, $i = \overline{1,n}$ în destinația B_j, $j = \overline{1,m}$ depinde de tipul de produs și de tipul de transport utilizat, acesta poate fi descris de o funcție concavă secvențial-liniară de forma:

$$\varphi_{ijkl}(x_{ijkl}) = \begin{cases} \Gamma_k * x_{ijkl}, & 0 \le x_{ijkl} \le \Delta_l, \\ \Gamma_k * \Delta_l, & x_{ijkl} > \Delta_l, \end{cases} \quad (3.4)$$

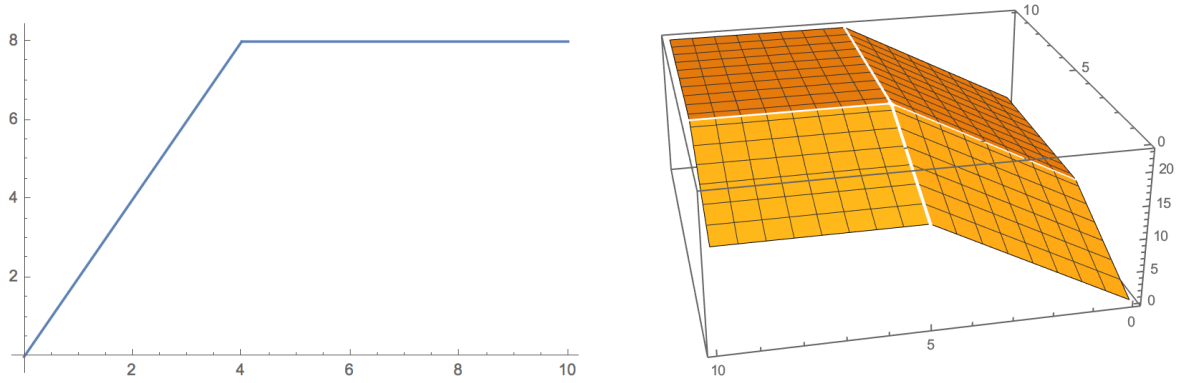
unde $i = \overline{1,n}$, $j = \overline{1,m}$, $k = \overline{1,p}$, $l = \overline{1,q}$, Δ_l este coeficientul care descrie dependența costului de tipul de transport utilizat, iar Γ_k – coeficientul care descrie dependența costului de tipul produsului transportat.

Când costul de transport depinde doar de tipul de transport utilizat, funcția de cost este descrisă astfel:

$$\varphi_{ijkl}(x_{ijkl}) = \begin{cases} x_{ijkl}, & 0 \le x_{ijkl} \le \Delta_l, \\ \Delta_l, & x_{ijkl} > \Delta_l, \end{cases} \quad (3.5)$$

unde $i = \overline{1,n}$, $j = \overline{1,m}$, $k = \overline{1,p}$, $l = \overline{1,q}$.

Deci, în ambele cazuri studiate, funcțiile de cost sunt funcții concave secvențial-liniare nedescrescătoare de forma dată în Figura 3.1 a):



a) b)

Fig. 3.1. Funcție secvențial-liniară (a) sumă a două funcții secvențial-liniare (b).
Sursa: elaborat de autor

Suma a două funcții de acest tip este o funcție neliniară, după cum este reprezentată și în Figura 3.1 b). În caz general, funcția obiectiv este dată de suma mai multor funcții concave secvențial-liniare. Se formulează o problemă a programării neliniare cu mulțimea soluțiilor admisibile descrisă de sistemul (3.2), care este mulțime convexă mărginită, iar funcția obiectiv este o funcție concavă secvențial-liniară. Soluția optimă este unul dintre punctele extreme ale mulțimii de restricții.

3.1.2 Algoritmul AME1

În cele ce urmează, soluționarea unei astfel de probleme presupune reduceri succesive a problemei neliniare formulate la probleme de programare liniară. Pentru soluționarea problemei liniare, care are aceleași restricții ca și problema neliniară, poate fi aplicată metoda simplex adaptată cazului problemei cu 4 indici, ca, de exemplu, cea propusă de A. Djamel et. al. (Djamel, Amel, Hoai, & Ahmed, 2012) sau R. Zitouni et. al. (Zitouni, Keraghel, & Benterki, 2007).

Algoritmul constă în reducerea consecutivă a problemei de optimizare neliniară la o serie de probleme de programare liniară. Pentru determinarea coeficienților care descriu funcția obiectiv liniară la care este redusă funcția obiectiv neliniară a problemei inițiale se folosește schema de soluționare a problemelor separabile deoarece ea satisface condițiile.

Algoritmul AME1

Pasul 1. Se construiește o soluție admisibilă inițială $x^0 = (x^0_{1111}, x^0_{1112}, \dots, x^0_{nmpq})$ a sistemului (3.2).

Pasul 2. Se determină valoarea funcției neliniare în x^0:

$$F(x^0) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p}\sum_{l=1}^{q}\varphi_{ijkl}\left(x^0_{ijkl}\right),$$

și se calculează valoarea coeficienților:

$$C_{ijkl} = \begin{cases} \dfrac{\varphi_{ijkl}\left(x^0_{ijkl}\right)}{x^0_{ijkl}}, & x^0_{ijkl} > 0, \\ \varphi'_{ijkl}(0), & x^0_{ijkl} = 0, \end{cases}$$

pentru fiecare (i, j, k, l).

Pasul 3. Se soluționează problema liniară de transport:

$$Z\left(x_{ijkl}\right) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p}\sum_{l=1}^{q} C_{ijkl}x_{ijkl} \to min,$$

cu restricțiile (3.2) și se obține soluția optimă $x^1 = (x^1_{1111}, x^1_{1112}, \dots, x^1_{nmpq})$.

Pasul 4. Se compară valorile $Z(x^1)$ și $F(x^0)$. Dacă $Z(x^1) < F(x^0)$ sau $Z(x^1) = F(x^0)$ și $x^1 \neq x^0$, atunci x^0 se substituie cu x^1 și se trece la *Pasul 2*, ceea ce înseamnă că se repornește procedura de liniarizare cu o altă soluție inițială. Dacă $Z(x^1) > F(x^0)$ sau $Z(x^1) = F(x^0)$ și $x^1 = x^0$, atunci soluția optimă a problemei neliniare este considerată valoarea $x^* = x^0$. **STOP.**

În cazul liniarizării funcției obiectiv, soluția optimă este dependentă de soluția admisibilă inițială utilizată pentru aproximare, deci trebuie aleasă cu mare grijă. În cazul problemelor de dimensiuni mari, este complicat de a realiza o alegere corectă, de aceea se utilizează o soluție aleatoare, ceea ce nu garantează obținerea soluției optime globale, ci doar a uneia pseudo optime.

Se formulează și se demonstrează următoarele teoreme:

Teorema 3.1. *Algoritmul AME1 converge către un optim local.*

Demonstrație. *Algoritmul AME1 pornește de la o soluție admisibilă care este unul dintre vârfurile poliedrului ce descrie mulțimea soluțiilor admisibile care satisfac sistemul de restricții liniare ale problemei neliniare cu funcții concave de cost. Soluția problemei liniare, care este problemă relaxată, satisface aceleași restricții, deci face parte din același domeniu de definiție și este unul dintre vârfurile poliedrului. Prin urmare, se poate spune că algoritmul converge către un optim local din domeniul de definiție a problemei inițiale neliniare.*□

Teorema 3.2. *Algoritmul AME1 necesită un volum de memorie de ordinul* $O(nmpq(n+m+p+q))$.

Demonstrație. *Graful* $G=(V,E)$ *este descris de o matrice de adiacență de dimensiunea* $nmpq\times(n+m+p+q)$. *Matricea care conține coeficienții liberi ai sistemului de restricții este de dimensiunea* $n+m+p+q$. *Soluția problemei este un tabel de dimensiunea* $nmpq$. *Prin urmare, implementarea algoritmului necesită memorie de ordinul* $O(nmpq(n+m+p+q))$.□

3.1.3 Implementarea și testarea algoritmului AME1

Sistemul Wolfram permite implementarea algoritmului, în care se pot utiliza un șir de funcții standard care simplifică semnificativ problema. Pentru obținerea soluției inițiale a algoritmului se aplică funcția standard *FindInstace[]* care soluționează sistemul de ecuații și, ca rezultat, livrează o soluție admisibilă din mulțimea de soluții ale sistemului de restricții. *LinearPrograming[]* este o funcție standard care permite soluționarea problemei de programare liniară.

Descrierea implementării algoritmului este prezentată în Anexa 9.

Rezultatele prezentate în Tabelele 3.1 și 3.2 sunt bazate pe probleme de diferite dimensiuni unde funcțiile de cost sunt funcții concave secvențial-liniare. Rețelele sunt descrise de n – surse, m – destinații, p – tipuri de produse transportate, q – tipuri de transport utilizat. Funcția de producere și consum presupune că totalul produselor disponibile în surse este egal cu totalul

produselor necesare în destinații egal cu 50 u. c. În același timp, totalul de tipuri de produse coincide cu capacitatea totală a tipurilor de transport care circulă în rețea și este de 50 u. c.

În Tabelul 3.1 poate fi vizualizat timpul de execuție a algoritmului AME1 pentru probleme de diferite dimensiuni, numărul de surse, destinații, tipuri de produse și tipuri de transport, pentru a căror soluționare sunt utilizate de la 24 de necunoscute pentru problema descrisă de 2 surse, 2 destinații, 2 tipuri de produse și 3 tipuri de transport până la 2401 necunoscute pentru problema descrisă de 7 surse, 7 destinații, 7 tipuri de produse și 7 tipuri de transport.

Tabelul 3.1. Timpul de execuție pentru algoritmul AME1. Sursa: elaborat de autor

n/m/p/q (secunde)	**2/2/2/3**	**2/2/3/3**	**2/3/3/3**	**3/3/3/3**	**3/3/3/4**	**3/3/4/4**	**3/4/4/4**	**4/4/4/4**
	0.0156	0.6250	0.1718	0.2031	0.3437	0.5312	0.8906	3.1093
	0.0155	0.6250	0.1093	0.2187	0.3125	0.5468	0.9062	3.1875
	0.0156	0.6250	0.1250	0.1875	0.3281	0.5312	0.9062	3.2968
	0.0156	0.0781	0.0937	0.2031	0.3125	0.5156	0.9218	3.2031
	0.0155	0.6125	0.1250	0.2031	0.3327	0.4843	0.8750	3.1562
Necunoscute	24	36	54	81	108	144	192	256
n/m/p/q (secunde)	**4/4/4/5**	**4/4/5/5**	**4/5/5/5**	**5/5/5/5**	**5/5/5/6**	**5/5/6/6**	**6/6/7/7**	**7/7/7/7**
	4.5000	6.7343	9.6250	15.3125	25.0781	33.3750	178.5310	474.141
	4.5937	6.9062	9.6406	15.2031	25.1719	33.1563	206.1250	483.000
	4.7187	6.7187	9.8593	15.0938	25.3750	33.7344	198.1553	477.814
	5.6093	6.7812	9.8125	15.2188	25.3438	33.6563	215.8750	490.328
	4.7500	6.9218	9.6718	15.3594	25.4063	33.6406	209.5278	486.512
Necunoscute	320	400	500	625	750	900	1764	2401

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

Timpul de execuție în acest caz depinde și de soluția admisibilă inițială obținută la *Pasul 1*, deoarece și numărul de iterații depinde de aceasta. În cazul problemelor de dimensiuni mari, nu se poate face o alegere a acestei soluții din insuficiență de timp care ar putea lua acest procedeu,

se operează cu o soluție admisibilă aleatoare. Astfel, algoritmul generează o soluție pseudo optimă a problemei formulate.

În Tabelul 3.2 sunt prezentate mai multe probleme de diferite dimensiuni pentru care valorile funcțiilor $F0$ și $Z1$ se modifică de la o iterație la alta. Analizând datele se observă că numărul de iterații necesar pentru obținerea soluției pseudo optime a problemei formulate nu depinde de dimensiunea problemei dar de soluția admisibilă inițială cu care începe procesul de reduceri succesive a problemei neliniare la probleme liniare.

Tabelul 3.2. Modificarea valorilor F0 și Z1 pentru algoritmul AME1.
Sursa: elaborat de autor

n/m/p/q / **Iterația**	**F0 și Z1** (u. c.)					
	4/4/4/4	**4/4/4/4**	**4/4/4/4**	**5/5/5/5**	**5/5/5/5**	**5/5/5/5**
I	27 / 20.80	28 / 19.91	29 / 23.76	34 / 26.87	33 / 22.36	34 / 27.9
II	20 / 16.70	18 / 16.47	22 / 19.5	19 / 18.94	20.6 / 19.30	22.66 / 21.25
III	14.5 / 14	15 / 13.91	15.5 / 15.1	18 / 17.4	17 / 16.83	21 / 20.96
IV	13.25 / 13.23	12.66 / 12.37	15 / 14.70	17.5 / 16.7	17 / 17	21 / 20.97
V	13.4 / 13.4	12 / 12	14.5 / 14.47	17 / 16.98	-	21 / 21
VI	-	12 / 12	14.3 / 14.3	16 / 16	-	-
VII	-	-	-	15.5 / 15.33	-	-
VIII	-	-	-	15 / 15	-	-
IX	-	-	-	15 / 14.83	-	-
X	-	-	-	14 / 14	-	-
	6/6/6/6	**6/6/6/6**	**6/6/6/6**	**7/7/7/7**	**7/7/7/7**	**7/7/7/7**
I	47 / 29.06	41 / 29.78	43 / 30.66	43 / 26.16	52 / 34.83	43 / 36.48
II	21 / 17.93	27.1 / 23.03	24 / 23.2	23.33 / 20.87	26.33 / 25.21	30.6 / 29,27
III	18 / 17.50	18.5 / 18.33	22.33 / 20.43	17 / 16.33	23 / 23	27 / 27.2
IV	16.75 / 15.8	18 / 18	16.66 / 16.5	16 / 16	23 / 22.6	27 / 27
V	15 / 14.39	17.16 / 16.91	16 / 16	-	21.5 / 21	26.5 / 26.38
VI	13 / 13	16 / 16	15 / 15	-	20 / 20	26 / 26

VII	13 / 13	-	-	-	20 / 20	-

De la o iterație la alta, valoarea $F0$ este mai mică sau egală decât valoarea $F0$ obținută la iterația precedentă. Cazul în care $F0$ rămâne nemodificat pe parcursul a câtorva iterații demonstrează încă o dată că problemele de acest fel pot avea mai multe minime locale. Aceste puncte permit diferite aproximări ale funcției neliniare cu una liniară, ceea ce, duce la o ieșire din blocaj fiind obținut un alt minim local cu valoarea funcției de cost mai mică.

3.1.4 Algoritmul genetic AMG1

În cele ce urmează este soluționată problema *PNT4I* aplicând un algoritm genetic care presupune codificarea problemei astfel încât fiecărui cromozom din populație i se poate asocia doar o soluție admisibilă. Fiecare populație nou-creată păstrează din populația precedentă cromozomii cu cele mai bune caracteristici, astfel încât să nu fie pierdute în timpul selecției soluțiile cărora le corespund costuri mai mici.

Deoarece fiecare cromozom trebuie să conțină informație despre fiecare indice care descrie problema, el este divizat în 3 compartimente astfel: *primul* conține lista de arce ordonată aleatoriu și descrie ordinea arcelor pe care este transportat produsul, *al doilea* conține o permutare a numerelor $k = \overline{1,p}$ asociate tipurilor de produse care trebuie transportate și descrie ordinea lor de transportare, *al treilea* conține o permutare a numerelor $l = \overline{1,q}$ asociate tipurilor de transport care pot fi utilizare și descrie ordinea lor de utilizare.

Algoritmul genetic AMG1 propus pentru soluționarea *PNT4I* cu funcții concave nedescrescătoare secvențial-liniare constă din următorii pași:

Pasul 1 *Inițializarea.* Se generează aleatoriu populația inițială din $4nm$ cromozomi după cum urmează: fiecare cromozom este descris de o listă de forma $P = \{P_{ij}, P_p, P_q\}$, unde $P_{ij} = \{p(i,j) \mid i = \overline{1,n}, j = \overline{1,m}\}$ – listă aleatoare de arce care descrie ordinea de transport a produselor pe arce, $P_p = \{p(1), p(2), \dots, p(p)\}$ – o permutare a numerelor $k = \overline{1,p}$ asociate tipurilor de produse, $P_q = \{p(1), p(2), \dots, p(q)\}$ – o permutare a numerelor $l = \overline{1,q}$ asociate tipurilor de transport.

Pasul 2 *Decodificarea și Evaluarea cromozomilor.* Decodificarea presupune că fiecărui cromozom i se asociază o soluție admisibilă de forma $x = (x_{1111}, x_{1112}, \dots, x_{nmpq})$ transportând fluxul din k produse cu cele l tipuri de transport din n surse către m destinații astfel:

Se creează tablourile unidimensionale care conțin respectiv cantitățile de produs în surse, necesitățile de produs în destinații, tipuri de produse cu cantitățile respective și tipurile de transport cu capacitățile respective $\alpha' = [\alpha'_1, \alpha'_2, \dots, \alpha'_n]$, $\beta' = [\beta'_1, \beta'_2, \dots, \beta'_m]$, $\gamma' = [\gamma'_1, \gamma'_2, \dots, \gamma'_p]$, $\delta' = [\delta'_1, \delta'_2, \dots, \delta'_q]$.

Pentru fiecare arc $(i,j) \in P_{ij}$, fiecare indice $k \in P_p$ și fiecare indice $l \in P_q$ se determină: $x_{ijkl} = min\{\alpha'_i,\ \beta'_j,\ \gamma'_k,\ \delta'_l\}$, după care are loc actualizarea $\alpha'_i := \alpha'_i - x_{ijkl}$, $\beta'_j := \beta'_j - x_{ijkl}$, $\gamma'_k := \gamma'_k - x_{ijkl}$, $\delta'_l := \delta'_l - x_{ijkl}$.

Evaluarea presupune determinarea valorii funcției obiectiv pentru fiecare dintre soluțiile obținute.

Pasul 3 *Selectarea cromozomilor.* Cromozomii sunt sortați în ordinea creșterii valorilor funcției obiectiv asociate cromozomului. Cromozomii din prima jumătate a populației sunt transferați în noua populație.

Pasul 4 *Încrucișarea cromozomilor.* Are loc între perechile de cromozomi transferați în populația $P(i)$ din populația $P(i-1)$ prin selecție. Aleatoriu este aplicată aceeași tăietură ambilor cromozomi-părinți asupra compartimentului 1, după care:

- primul cromozom-urmaș primește consecutivitatea arcelor de până la tăietură a cromozomului-mamă și ordinea de distribuire a tipurilor de produse, iar celelalte arce (din a doua parte a compartimentului 1) păstrează ordinea arcelor cromozomului-tată și ordinea de utilizare a tipurilor de transport;
- al doilea cromozom-urmaș primește consecutivitatea arcelor de după tăietură a cromozomului-tată și ordinea de distribuire a tipurilor de produse, iar celelalte arce (din prima parte a compartimentului 1) păstrează ordinea arcelor cromozomului-mamă și ordinea de utilizare a tipurilor de transport.

Pasul 5 *Mutația.* Unei gene a cromozomului-urmaș i se aplică o mutație care are loc cu o probabilitate $\varepsilon \in [0.1, 0.5]$ și presupune:

- pentru fiecare compartiment *doi* – tipuri de produse, *trei* – tipuri de transport, aleatoriu se decide dacă este aplicată mutația;
- în fiecare dintre compartimentele alese se schimbă valorile între două elemente alese aleatoriu.

Pasul 6 *Verificarea condiției de oprire.* Implică oprirea algoritmului după *k* iterații. În calitate de soluție a problemei servește soluția care corespunde cromozomului cu valoarea funcției obiectiv minimă din ultima populație construită. Se trece la *Pasul 2* dacă condiția de oprire nu este satisfăcută.

Remarca 3.1. *În cazul problemelor de dimensiuni mari, drept condiție de oprire a execuției algoritmului poate servi și durata de execuție în timp. Deci, după o perioadă anumită de timp generarea noilor populații este stopată, iar ca soluție a problemei este acceptată soluția care corespunde cromozomului cu cele mai bune caracteristici din ultima populație.*

Observația 3.1. *Algoritmul AMG1 se aplică în cazul problemelor de transport care îndeplinesc condiția de echilibru (3.3).*

Remarca 3.2. *Algoritmul AMG1 se aplică în cazul rețelelor ce îndeplinesc următoarele condiții:*

a. *Graful rețelei de transport este conex și aciclic;*
b. *Există cel puțin 2 surse, 2 destinații, 2 tipuri de produse și 2 tipuri de transport;*
c. *Vârfurile surse nu conțin arce ascendente, iar vârfurile destinații nu conțin arce descendente.*

Exemplul (codificare) 3.1. Se consideră rețeaua de transport, descrisă de graful din Figura 3.2, cu sursele A_1 și A_2 care conțin respectiv $\alpha_1 = 20$ u. c. și $\alpha_2 = 80$ u. c. de produs de tipul P_1 și P_2 respectiv câte $\gamma_1 = 50$ u.c. și $\gamma_2 = 50$ u. c. fiecare. Produsele sunt transportate cu ajutorul tipurilor de transport T_1 și T_2 de capacitatea respectivă $\delta_1 = 60$ u.c. și $\delta_2 = 40$ u. c. în destinațiile B_1 și B_2 care au necesitățile respective $\beta_1 = 70$ u. c. și $\beta_2 = 30$ u.c..

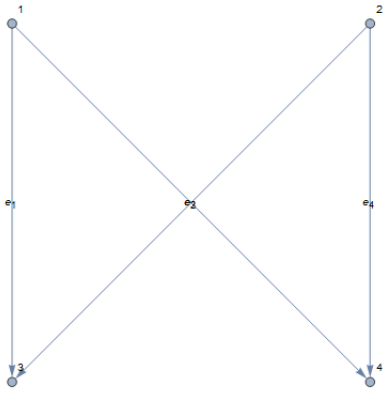


Fig. 3.2. Rețea cu două surse și două destinații.
Sursa: elaborat de autor

Un cromozom a populației, în acest caz este descris de:

• *compartimentul 1* care conține lista arcelor $\{(1,1), (2,2), (1,2), (2,1)\}$;

• *compartimentul 2* care conține ordinea de deservire a tipurilor de produse, fie $\{1,2\}$.

• *compartimentul 3* care conține ordinea de utilizare a tipurilor de transport, fie $\{2,1\}$.

Astfel, se obține un cromozom de forma:

$$\{\{(1,1), (2,2), (1,2), (2,1)\},\{1,2\},\{2,1\}\}.$$

Exemplul (decodificare) 3.2. Pentru cromozomul generat se construiește soluția admisibilă de forma $x = (x_{1111}, x_{1112}, x_{1121}, x_{1122}, x_{1211}, x_{1212}, x_{1221}, x_{1222}, x_{2111}, x_{2112}, x_{2121}, x_{2122}, x_{2211}, x_{2212}, x_{2221}, x_{2222})$ astfel:

Se creează tablourile $\alpha' = [20, 80]$, $\beta' = [70,30]$, $\gamma' = [50,50]$, $\delta' = [60,40]$. După care elementele soluției se calculează după cum urmează:

- $x_{1112} = min\{20,70,50,40\} = 20,\ \alpha_1' \coloneqq 20 - 20 = 0,\ \beta_1' \coloneqq 70 - 20 = 50, \gamma_1' \coloneqq 50 - 20 = 30,\ \delta_2' \coloneqq 40 - 20 = 20;$
- $x_{1111} = min\{0,50,30,60\} = 0, \alpha_1' \coloneqq 0, \beta_1' \coloneqq 50, \gamma_1' \coloneqq 30,\ \delta_1' \coloneqq 60;$
- $x_{1122} = min\{0,50,50,20\} = 0, \alpha_1' \coloneqq 0, \beta_1' \coloneqq 50, \gamma_2' \coloneqq 50,\ \delta_2' \coloneqq 20;$
- $x_{1121} = min\{0,50,50,60\} = 0, \alpha_1' \coloneqq 0, \beta_1' \coloneqq 50, \gamma_2' \coloneqq 50,\ \delta_1' \coloneqq 60;$
- $x_{2212} = min\{80,30,30,20\} = 20, \alpha_2' \coloneqq 80 - 20 = 60, \beta_2' \coloneqq 30 - 20 = 10, \gamma_1' \coloneqq 30 - 20 = 10,\ \delta_2' \coloneqq 20 - 20 = 0;$
- $x_{2211} = min\{60,10,10,60\} = 10,\ \alpha_2' \coloneqq 60 - 10 = 50,\ \beta_2' \coloneqq 10 - 10 = 0, \gamma_1' \coloneqq 10 - 10 = 0,\ \delta_1' \coloneqq 60 - 10 = 50;$
- $x_{2222} = min\{50,00,50,0\} = 0, \alpha_2' \coloneqq 50, \beta_2' \coloneqq 0, \gamma_2' \coloneqq 50,\ \delta_2' \coloneqq 0;$
- $x_{2221} = min\{50,0,50,50\} = 0, \alpha_2' \coloneqq 50, \beta_2' \coloneqq 0, \gamma_2' \coloneqq 50,\ \delta_1' \coloneqq 50;$
- $x_{1212} = min\{0,0,0,0\} = 0, \alpha_1' \coloneqq 0, \beta_2' \coloneqq 0, \gamma_1' \coloneqq 0,\ \delta_2' \coloneqq 0;$
- $x_{1211} = min\{0,0,0,50\} = 0, \alpha_1' \coloneqq 0, \beta_2' \coloneqq 0, \gamma_1' \coloneqq 0,\ \delta_1' \coloneqq 50;$
- $x_{1222} = min\{0,0,50,0\} = 0, \alpha_1' \coloneqq 0, \beta_2' \coloneqq 0, \gamma_2' \coloneqq 50,\ \delta_2' \coloneqq 0;$
- $x_{1221} = min\{0,0,50,50\} = 0, \alpha_1' \coloneqq 0, \beta_2' \coloneqq 0, \gamma_2' \coloneqq 50,\ \delta_1' \coloneqq 50;$
- $x_{2112} = min\{50,50,0,0\} = 0, \alpha_2' \coloneqq 50, \beta_1' \coloneqq 50, \gamma_1' \coloneqq 0,\ \delta_2' \coloneqq 0;$
- $x_{2111} = min\{50,50,0,50\} = 0, \alpha_2' \coloneqq 50, \beta_1' \coloneqq 50, \gamma_1' \coloneqq 0,\ \delta_1' \coloneqq 50;$
- $x_{2122} = min\{50,50,50,0\} = 0, \alpha_2' \coloneqq 50, \beta_1' \coloneqq 50, \gamma_2' \coloneqq 50,\ \delta_2' \coloneqq 0;$
- $x_{2121} = min\{50,50,50,50\} = 50,\ \ \alpha_1' \coloneqq 50 - 50 = 0,\ \ \beta_1' \coloneqq 50 - 50 = 0, \gamma_2' \coloneqq 50 - 50 = 0,\ \delta_1' \coloneqq 50 - 50 = 0.$

  Se obține $x = (0,20,0,0,0,0,0,0,0,0,50,0,10,20,0,0)$.

Observația 3.2. *În cazul implementării practice a algoritmului AMG1 lista de arce poate fi descrisă și prin numărul arcului din lista tuturor acelor care aparțin grafului. Acest fapt nu reduce din necesarul total de memorie, dar permite implementarea mai simplă a algoritmului.*

Fie mulțimea arcelor $\{(1,1),(1,2),(2,1)(2,2)\}$ care descrie primul compartiment al unui cromozom. Această listă poate fi înlocuită cu lista $\{1,2,3,4\}$ și cromozomul $\{\{(1,2)(1,1)(2,2)(2,1)\}|\{1,2\}|\{2,1\}\}$ ia forma $\{\{2,1,3,4\}|\{1,2\}|\{2,1\}\}$.

Exemplul (încrucișare) 3.3. Pentru o rețea cu 2 surse, 2 destinații, 3 tipuri de produse și 3 tipuri de transport care conține 4 arce sunt dați 2 cromozomi-părinți care participă la încrucișare. Tăietura este aplicată la mijlocul mulțimii de arce din compartimentul 1:

Cromozom-mama {{1,4,2,3},{1,2,3},{1,3,2}}

Cromozom-tata {{4,1,3,2},{3,2,1},{3,1,2}}

atunci cromozomii-urmași sunt:

Cromozom-urmaș1 {{1,4,3,2},{1,2,3},{3,1,2}}

Cromozom-urmaș2 {{1,4,3,2},{3,2,1},{1,3,2}}.

Exemplul (mutație) 3.4. În exemplul de mai jos este selectat aleatoriu compartimentul tipurilor de transport asupra căruia se aplică mutația:

Cromozom-urmaș {{1,4,3,2},{1,2,3},{3,1,2}}

pentru care au fost schimbate cu locul două elemente după cum urmează:

Cromozom-urmaș {{1,4,3,2},{1,2,3},{2,1,3}}.

Se formulează și se demonstrează următoarele teoreme:

Teorema 3.3. *Algoritmul AMG1 necesită un volum de memorie de ordinul* $O(nm(n+m+p+q))$.

Demonstrație. *Datele de intrare descrise de tabelul de restricții sunt de dimensiunea* $nmpq$. *Pentru păstrarea soluției asociate cromozomului cu cele mai bune caracteristici din populație este necesară* $nmpq$ *memorie. Păstrarea unui cromozom necesită* $nm+p+q$ *memorie iar pentru întreaga populație –* $nm(nm+p+q)$ *memorie. Prin urmare, algoritmul AMG1 necesită un volum de memorie de ordinul* $O(nm(nm+p+q))$.□

Teorema 3.4. *Complexitatea unei iterații a algoritmului AMG1 este de ordinul* $O(n^2m^2pq)$.

Demonstrație. *Pentru inițializarea datelor de intrare descrise de tabelul de restricții sunt necesare* $O(nmpq)$ *operații. Generarea unui cromozom necesită* $O(nm+p+q)$ *operații, astfel încât generarea întregii populații necesită* $O(nm(nm+p+q))$ *operații. Evaluarea soluției asociate unui cromozom necesită* $O(nmpq)$ *operații. Deoarece în populație sunt* nm *cromozomi, complexitatea de evaluare a tuturor cromozomilor este* $O(n^2m^2pq)$. *Încrucișarea este de complexitate* $O(nm+p+q)$ *pentru un cromozom și de complexitate* $O(nm(nm+p+q))$ *pentru întreaga populație. Prin urmare, o iterație a algoritmului AMG1 are complexitatea de ordinul* $O(n^2m^2pq)$.□

Observația 3.3. *Din Teorema 3.4 rezultă că timpul de execuție a algoritmului AMG1 este* $O(Un^2m^2pq)$, *unde* U *este numărul de iterații necesar pentru obținerea unei soluții asociate cromozomului cu caracteristici bune.*

Teorema 3.5. *Algoritmul AMG1 converge către un optim local.*

Demonstrație. *Din populația* $P(i-1)$ *sunt transferați în* $P(i)$ *cromozomii pentru care valoarea funcției obiectiv în soluția asociată este minimă. Cromozomilor populațiilor construite la fiecare iterație le este asociată câte o soluție admisibilă. Se poate spune că după executarea unui număr finit de iterații este depistat cromozomul căruia îi este asociată soluția ce descrie optimul local. Prin urmare, algoritmul AMG1 converge către un optim local.*□

3.1.5 Implementarea și testarea algoritmului AMG1

Algoritmul AMG1 este implementat în limbajul Wolfram și testat în baza unui șir de exemple, probleme de transport de diferite dimensiuni în ceea ce privește numărul de surse, numărul de destinații, numărul tipurilor de produse transportate și numărul tipurilor de transport care pot fi utilizate.

Un exemplu practic de aplicare a algoritmului AMG1 pentru soluționarea problemei neliniare de transport pe rețea descrisă de surse, destinații, tipuri de produse și tipuri de transport este prezentat în Anexa 10. Codul scris în Wolfram Mathematica poate fi accesat pe https://github.com/TatianaPasa/GeneticAlgorithm.

Tabelele 3.3 și 3.4 conțin rezultatele soluționării unui șir de probleme de diferite dimensiuni, unde funcțiile de cost sunt funcții concave secvențial-liniare. Problemele de transport soluționate sunt descrise de n – surse, m – destinații, p – tipuri de produse transportate, q – tipuri de transport utilizate. În baza funcției de producere și consum totalul produselor disponibile în surse este egal cu totalul produselor necesar în destinații și este egal cu 100 u. c. Pe de altă parte, totalul de tipuri de produse coincide cu capacitatea totală a tipurilor de transport utilizat și este de 100 u. c.

Tabelul 3.3. Timpul de execuție pentru algoritmul AMG1. Sursa: elaborat de autor

n/m/p/q (secunde)	**3/3/3/3**	**3/3/3/4**	**3/3/4/4**	**3/4/4/4**	**4/4/4/4**	**4/4/4/5**	**4/4/5/5**	**4/5/5/5**
	0.4843	0.6562	0.8437	1.4843	2.5781	3.1562	3.8125	6.2500
	0.4531	0.5625	0.8281	1.5000	2.5312	3.1406	3.9062	6.1406
	0.4843	0.5937	0.7968	1.5156	2.5468	3.1093	3.8906	6.2031
	0.4843	0.6250	0.8125	1.4687	2.4843	3.1250	3.9218	6.1406
	0.4843	0.6093	0.8750	1.4375	2.5781	3.0000	3.8281	6.1250
Necunoscute	81	108	144	192	256	320	400	500
n/m/p/q	**5/5/5/5**	**5/5/5/6**	**5/5/6/6**	**6/6/7/7**	**7/7/7/7**	**8/8/8/8**	**9/9/9/9**	**10/10/10/10**

(secunde)								
	9.3593	11.3125	13.7188	38.8438	72.8125	159.375	334.359	597.641
	9.4218	11.2031	13.6250	39.4063	74.4219	157.547	331.313	598.750
	9.5937	11.2813	13.5313	39.4688	72.2031	159.969	316.266	597.703
	9.4834	11.1875	13.3906	39.7813	71.3125	159.797	316.016	598.531
	9.4218	11.3438	13.4531	39.5156	71.2500	162.203	317.766	604.328
Necunoscute	625	750	900	1764	2401	4096	6561	10000

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

În Tabelul 3.3 este dat timpul de execuție pentru algoritmul AMG1 în cazul problemelor de diferite dimensiuni: numărul de surse, destinații, tipuri de produse și tipuri de transport, pentru a căror soluționare sunt utilizate de la 81 de necunoscute pentru problema cu 3 surse, 3 destinații, 3 tipuri de produse și 3 tipuri de transport până la 10000 de necunoscute pentru problema cu 10 surse, 10 destinații, 10 tipuri de produse și 10 tipuri de transport. După cum se observă, algoritmul permite obținerea unor soluții pseudo optime în timp rezonabil chiar și în cazul problemelor de transport de dimensiuni mari.

În Tabelul 3.4 sunt prezentate rezultatele soluționării unui șir de probleme de transport de dimensiuni diferite, unde se poate observa cum pentru fiecare din cele 10 populații generate se modifică valoarea funcției obiectiv calculată pentru soluția asociată cromozomului cu cele mai bune caracteristici din populația respectivă.

Tabelul 3.4. Modificarea valorilor $F(x)$ și $F_T(x)$ pentru algoritmul AMG1. Sursa: elaborat de autor

Iterația	**n/m/p/q** (u. c.)				
	4/4/4/4	**5/5/5/5**	**6/6/6/6**	**7/7/7/7**	**10/10/10/10**
I	19 / 1564	21 / 3253	27 / 5718	33 / 9772	52 / 27616
II	18 / 1507	18 / 2978	27 / 5433	30 / 9477	47 / 26788
III	16 / 1473	18 / 2852	27 / 5255	30 / 9322	47 / 26191
IV	16 / 1374	18 / 2741	24 / 5034	30 / 9099	47 / 25761
V	16 / 1365	18 / 2704	24 / 4982	30 / 8756	47 / 25455
VI	16 / 1291	17 / 2552	24 / 4861	30 / 8574	47 / 25140
VII	16 / 1255	16 / 2390	24 / 4612	30 / 8374	47 / 24877
VIII	16 / 1215	16 / 2364	21 / 4510	30 / 8304	44 / 24570
IX	14 / 1160	16 / 2229	21 / 4340	24 / 8127	44 / 24346

X	14 / 1118	14 / 2001	21 / 4209	24 / 7848	41 / 23941

Analizând datele din acest tabel se observă că $F_T(x)$ întotdeauna descreşte de la o iteraţie la alta, ceea ce presupune că fiecare populaţie nouă conţine mai mulţi cromozomi cu caracteristici mai bune decât cei din populaţia anterioară, deci şi costuri de transport mai mici. Deşi se vede că la câteva iteraţii consecutive se repetă aceeaşi valoare $F(x)$ se atrage atenţia la faptul că algoritmul permite ieşirea dintr-un astfel de blocaj, într-o soluţie locală, fiind generat un nou cromozom cu caracteristici mai bune, după câteva iteraţii, căruia i se asociază o soluţie pentru care valoarea funcţiei obiectiv este mai mică.

Soluţia problemei obţinută după un număr stabilit de iteraţii este o soluţie optimă locală care se apropie de cea globală cu atât mai mult cu cât mai multe iteraţii sunt executate. Numărul de iteraţii depinde de dimensiunea problemei soluţionate, de timpul disponibil avut pentru a furniza soluţia dar şi de capacităţile tehnologice de care dispunem.

Dacă aceleaşi probleme neliniare de transport cu 4 indici se soluţionează aplicând de câteva ori algoritmii AME1 şi AMG1, se ajunge la concluzia că rezultatul furnizat depinde în mare parte de pasul iniţial. Deşi algoritmul AMG1 depinde de populaţia iniţială, după un număr suficient de iteraţii el permite obţinerea soluţiei optime locale din vecinătatea optimului global, deci a unei soluţii pseudo optime. O situaţie mai dificilă este în cazul algoritmului AME1, pentru care doar o alegere corectă a soluţiei iniţiale permite obţinerea soluţiei optime locale din vecinătatea optimului global, fapt ce nu poate fi realizat în special pentru probleme de dimensiuni mari. O caracteristică importantă a lui AMG1 este ieşirea cu succes din blocaje în minime locale şi, astfel, îmbunătăţirea soluţiei ceea ce presupune un cost de transport mai mic.

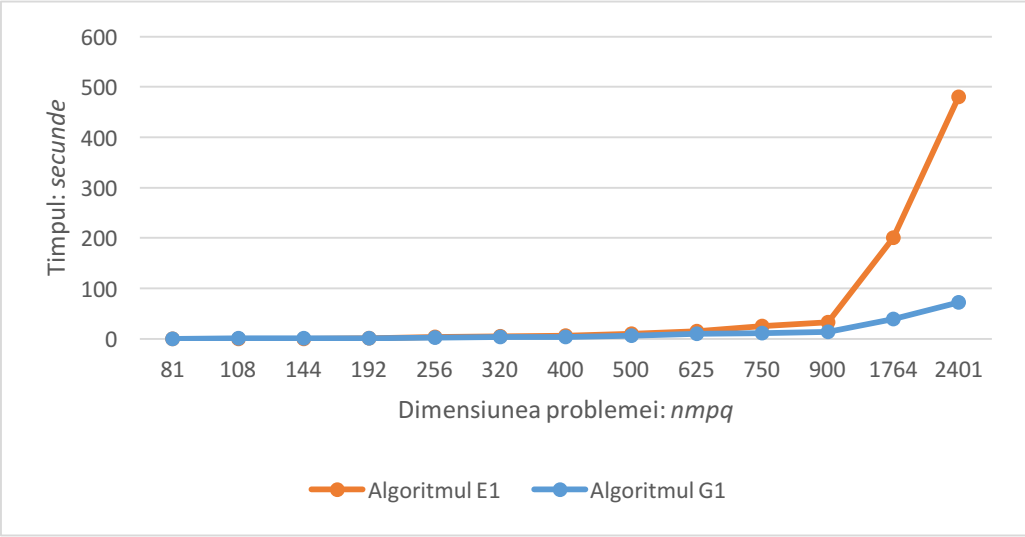


Fig. 3.3. Timpul de execuţie pentru AME1 şi AMG1. Sursa: elaborat de autor

Din Figura 3.3 se observă că algoritmul AMG1 furnizează soluție, după 10 iterații, în timp de cca un minut chiar și pentru probleme cu peste 2500 de necunoscute. În cazul implementării algoritmului AME1, când numărul necunoscutelor trece de o mie timpul de furnizare a soluției crește semnificativ. Pentru ambii algoritmi experimental este demonstrată convergența către soluții optime locale în timp rezonabil.

3.2 Problema de transport pe rețea cu mai mulți indici

3.2.1 Formularea problemei. Proprietăți

În cele ce urmează, se consideră problema descrisă de:

- n surse $A_1, A_2, \dots, A_n$ care posedă cantitățile de produs respectiv $\alpha_1, \alpha_2, \dots, \alpha_n$;
- m destinații $B_1, B_2, \dots, B_m$ cu necesitățile respective de produs $\beta_1, \beta_2, \dots, \beta_m$;
- p tipuri de produse $P_1, P_2, \dots, P_p$ de cantitățile respective $\gamma_1, \gamma_2, \dots, \gamma_p$;
- q tipuri de transport $T_1, T_2, \dots, T_q$ cu capacitățile respective $\delta_1, \delta_2, \dots, \delta_q$;
- g noduri (puncte) intermediare $v_r \in V_{int}$ în care nu are loc nici producerea și nici consumul din cantitatea de produs.

Costul de transport pe rețea este descris de o funcție neliniară $\varphi(x)$, pentru fiecare arc $e \in E$, care depinde de volumul fluxului de produs transportat. Scopul problemei este minimizarea costului total de transport al tuturor tipurilor de produse disponibile în surse pentru aprovizionarea tuturor destinațiilor cu necesarul respectiv utilizând toate tipurile de transport cu capacități diferite disponibile în rețea.

În acest caz, rețeaua de transport este descrisă de un graf orientat aciclic $G = (V, E)$, cu mulțimea de vârfuri $V = V_s \cup V_t \cup V_{int}$, unde V_s – mulțimea de surse, $|V_s| = n$, V_t – mulțimea de destinații $|V_t| = m$ și V_{int} – mulțimea de puncte intermediare $|V_{int}| = g$ iar E – mulțimea de arce, $|E| = u$. Această problemă face parte din grupul de probleme neliniare de transport cu 5 indici descrisă de surse, destinații, puncte intermediare, tipuri de produse și tipuri de transport.

Pentru a simplifica formularea problemei, dar și descrierea/implementarea algoritmului de soluționare, problema neliniară de transport descrisă de surse, destinații, tipuri de produse, tipuri de transport și puncte intermediare este formulată ca o problemă neliniară de transport pe rețea cu 3 indici (PNTR3I) descrisă de arce, tipuri de produse și tipuri de transport.

În aceste condiții, pentru determinarea costului minim de transport trebuie respectate restricțiile de conservare a fluxului de produs în rețea:

1) Cantitatea totală de produs de tipul P_k, $k = \overline{1,p}$ în mărime de $\gamma_k, k = \overline{1,p}$, care iese din toate sursele rețelei este descrisă de relația:

$$\sum_{e \in E^-(V_s)} \sum_{l=1}^{q} x_{ekl} = \gamma_k, \qquad k = \overline{1,p}, \tag{3.6}$$

unde $E^-(V_s)$ – mulțimea arcelor care ies din sursele rețelei.

2) Cantitatea totală de transport de tipul T_l , $l = \overline{1,q}$ cu capacitatea $\delta_l, l = \overline{1,q}$ care iese din toate sursele rețelei este descrisă de relația:

$$\sum_{e \in E^-(V_s)} \sum_{k=1}^{p} x_{ekl} = \delta_l, \qquad l = \overline{1,q}, \tag{3.7}$$

unde $E^-(V_s)$ – mulțimea arcelor care ies din sursele rețelei.

3) Cantitatea totală de produs care iese din fiecare sursă A_i, $i = \overline{1,n}$ este α_i, $i = \overline{1,n}$ și descrisă de relația:

$$\sum_{e \in E^-(s_i)} \sum_{k=1}^{p} \sum_{l=1}^{q} x_{ekl} = \alpha_i, \qquad i = \overline{1,n}, \tag{3.8}$$

unde $s_i \in V_s$ – sursa i a rețelei, $E^-(s_i)$ – mulțimea arcelor care ies din sursa i a rețelei.

4) Cantitatea totală de produs care intră în fiecare destinație B_j, $j = \overline{1,m}$ este β_j, $j = \overline{1,m}$ și descrisă de relația:

$$\sum_{e \in E^+(d_j)} \sum_{k=1}^{p} \sum_{l=1}^{q} x_{ekl} = \beta_j, \qquad j = \overline{1,m}, \tag{3.9}$$

unde $d_j \in V_t$ – destinația j a rețelei, $E^+(d_j)$ – mulțimea arcelor care intră în destinația j a rețelei.

5) Cantitatea de produs $\gamma_k, k = \overline{1,p}$ care intră într-un vârf intermediar $v_r \in V_{int}$ coincide cu cantitatea de produs care iese din acel vârf:

$$\sum_{e \in E^+(v_r)} \sum_{l=1}^{q} x_{ekl} - \sum_{e \in E^-(v_r)} \sum_{l=1}^{q} x_{ekl} = 0, \qquad r = \overline{1,g}, k = \overline{1,p}, \tag{3.10}$$

unde $v_r \in V_{int}$ – vârf intermediar al rețelei, $E^-(v_r)$ – mulțimea arcelor care ies din punctul intermediar r a rețelei, $E^+(v_r)$ – mulțimea arcelor care intră în punctul intermediar r a rețelei.

6) Cantitatea de transport $\delta_l, l = \overline{1,q}$ care intră într-un vârf intermediar $v_r \in V_{int}$ coincide cu cantitatea de transport care iese din acel vârf:

$$\sum_{e\in E^+(v_r)}\sum_{k=1}^{p} x_{ekl} - \sum_{e\in E^-(v_r)}\sum_{k=1}^{p} x_{ekl} = 0, \qquad r=\overline{1,g}, l=\overline{1,q}, \tag{3.11}$$

unde $v_r \in V_{int}$ – vârf intermediar al rețelei, $E^-(v_r)$ – mulțimea arcelor care ies din punctul intermediar r a rețelei, $E^+(v_r)$ – mulțimea arcelor care intră în punctul intermediar r a rețelei.

7) Valorile care descriu fluxul de produs pe arcele rețelei sunt mărimi pozitive:

$$x_{ekl} \geq 0, \qquad \text{pentru fiecare } (e,k,l). \tag{3.12}$$

PNTR3I constă în determinarea unui flux x^* care minimizează funcția de cost:

$$F(x) = \sum_{e\in E}\sum_{k=1}^{p}\sum_{l=1}^{q} \varphi_{ekl}(x_{ekl}), \tag{3.13}$$

unde $\varphi_{ekl}(x_{ekl})$ sunt funcții concave secvențial-liniare nedescrescătoare de cost, adică se cere soluționarea problemei neliniare:

$$F(x) \to min, \tag{3.14}$$

$$\begin{cases} \sum_{e\in E^-(V_s)}\sum_{l=1}^{q} x_{ekl} = \gamma_k, \quad k=\overline{1,p}, \\ \sum_{e\in E^-(V_s)}\sum_{k=1}^{p} x_{ekl} = \delta_l, \quad l=\overline{1,q}, \\ \sum_{e\in E^-(s_i)}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ekl} = \alpha_i, \quad i=\overline{1,n}, \\ \sum_{e\in E^+(d_j)}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ekl} = \beta_j, \quad j=\overline{1,m}, \\ \sum_{e\in E^+(v_r)}\sum_{l=1}^{q} x_{ekl} - \sum_{e\in E^-(v_r)}\sum_{l=1}^{q} x_{ekl} = 0, \quad r=\overline{1,g}, k=\overline{1,p}, \\ \sum_{e\in E^+(v_r)}\sum_{k=1}^{p} x_{ekl} - \sum_{e\in E^-(v_r)}\sum_{k=1}^{p} x_{ekl} = 0, \quad r=\overline{1,g}, l=\overline{1,q}, \\ x_{ekl} \geq 0, \quad \forall (e,k,l), \end{cases} \tag{3.15}$$

pentru care sunt respectate condițiile de pozitivitate, deci mărimile $\alpha_1, \alpha_2, \dots, \alpha_n$, $\beta_1, \beta_2, \dots, \beta_m$, $\gamma_1, \gamma_2, \dots, \gamma_p$ și $\delta_1, \delta_2, \dots, \delta_q$ sunt pozitive și nenule.

Forma generală a soluției optime a *PNTR3I* are forma $x^* = (x^*_{111}, x^*_{112}, \dots, x^*_{upq})$ pentru o rețea descrisă de u arce. Se presupune că toate datele sunt valori reale și se dorește să se obțină ca soluție optimă o valoare reală fezabilă.

3.2.2 Algoritmul AME2

În cele ce urmează se propune algoritmul AME2 de soluționare a problemei formulate care presupune reduceri succesive a problemei neliniare la probleme liniare pentru care se cunosc metode de soluționare.

Algoritmul constă în reducerea consecutivă a problemei de optimizare neliniară la o serie de probleme de programare liniară. Pentru determinarea coeficienților care descriu funcția obiectiv liniară la care este redusă funcția obiectiv neliniară a problemei inițiale se folosește schema de soluționare a problemelor separabile deoarece ea satisface condițiile.

Algoritmul AME2

Pasul 1. Se construiește o soluție admisibilă inițială $x^0 = (x^0_{111}, x^0_{112}, \dots, x^0_{upq})$ a sistemului de restricții (3.15).

Pasul 2. Se determină valoarea funcției în x^0:

$$F(x^0) = \sum_{e\in E}\sum_{k=1}^{p}\sum_{l=1}^{q} \varphi_{ekl}(x^0_{ekl}),$$

și se calculează valoarea coeficienților:

$$C_{ekl} = \begin{cases} \dfrac{\varphi_{ekl}(x^0_{ekl})}{x^0_{ekl}}, & x^0_{ekl} > 0, \\ \varphi'_{ekl}(0), & x^0_{ekl} = 0, \end{cases}$$

pentru fiecare (e, k, l).

Pasul 3. Se soluționează problema liniară de transport:

$$Z(x_{ekl}) = \sum_{e\in E}\sum_{k=1}^{p}\sum_{l=1}^{q} C_{ekl}x_{ekl} \to min,$$

cu restricțiile (3.15) și se obține soluția optimă $x^1 = (x^1_{111}, x^1_{112}, \dots, x^1_{upq})$ pentru o rețea descrisă de u arce.

Pasul 4. Se compară valorile $Z(x^1)$ și $F(x^0)$. Dacă $Z(x^1) < F(x^0)$ sau $Z(x^1) = F(x^0)$ și $x^1 \neq x^0$ atunci x^0 se substituie cu x^1 și se trece la *Pasul 2* ceea ce înseamnă că se repornește procedura de liniarizare cu o altă soluție inițială. Dacă $Z(x^1) > F(x^0)$ sau $Z(x^1) = F(x^0)$ și $x^1 = x^0$, atunci soluția optimă a problemei neliniare este considerată valoarea $x^* = x^0$, se păstrează mărimea $F(x^0)$ și $x^* = x^0$ care îi corespunde și **STOP**.

În cazul liniarizării funcției obiectiv soluția optimă depinde de soluția admisibilă inițială utilizată pentru aproximare. Pentru probleme de dimensiuni mari este complicat de ales o soluție

potrivită care ar garanta obținerea optimului global. Din acest motiv, pentru soluționarea problemei este utilizată o soluție admisibilă aleatoare care permite generarea unei soluții pseudo optime.

În cele ce urmează, se formulează și se demonstrează următoarele teoreme:

Teorema 3.6. *Algoritmul AME2 converge către un optim local.*

Demonstrație. *Algoritmul AME2 pornește de la o soluție admisibilă care este unul dintre vârfurile poliedrului ce descrie mulțimea soluțiilor admisibile care satisfac sistemul de restricții liniare ale problemei neliniare cu funcții concave de cost. Soluția problemei liniare, care este problemă relaxată, satisface aceleași restricții, deci face parte din domeniul de definiție și este unul dintre vârfurile poliedrului. Prin urmare, se poate spune că algoritmul converge către un optim local din domeniul de definiție a problemei inițiale neliniare.*□

Teorema 3.7. *Algoritmul AME2 necesită un volum de memorie de ordinul* $O(upq(n + m + g(p + q)))$.

Demonstrație. *Graful* $G = (V, E)$ *este descris de o matrice de adiacență de dimensiunea* $upq(n + m + 2(p + q) + g(p + q))$. *Matricea care conține coeficienții liberi este de dimensiunea* $n + m + 2(p + q) + g(p + q)$. *Soluția problemei este un tabel de dimensiunea* upq. *Ca urmare, implementarea algoritmului necesită memorie de ordinul* $O(upq(n + m + g(p + q)))$.□

3.2.3 Implementarea și testarea algoritmului AME2

Sistemul Wolfram permite utilizarea unui șir de funcții standard care simplifică implementarea algoritmului. Un exemplu de implementare a algoritmului AME2 este prezentat în Anexa 11.

Rezultatele prezentate în Tabelele 3.5 și 3.6 sunt bazate pe probleme de diferite dimensiuni unde funcțiile de cost sunt funcții concave secvențial-liniare. Rețelele sunt descrise de n – surse, m – destinații, p – tipuri de produse transportate, q – tipuri de transport utilizate și g – puncte intermediare. Conform funcției de producere și consum, totalul produselor disponibile în surse este egal cu totalul produselor necesare în destinații și este egal cu 50 u. c. Se cunoaște și, totalul de tipuri diferite de produse coincide cu capacitatea totală a tipurilor de transport din rețea și este egală cu 50 u. c.

În Tabelul 3.5 este dat timpul de execuție a algoritmului AME2 pentru probleme de diferite dimensiuni, pentru a căror soluționare sunt utilizate de la 48 de necunoscute pentru problema cu 2

surse, 2 destinații, 2 tipuri de produs, 2 tipuri de transport și 2 puncte intermediare până la 4428 de necunoscute pentru problema cu 6 surse, 6 destinații, 6 tipuri de produs, 6 tipuri de transport și 6 puncte intermediare.

Tabelul 3.5. Timpul de execuție pentru algoritmul AME2. Sursa: elaborat de autor

n/m/p/q/g (secunde)	**2/2/2/2/2**	**2/2/3/3/2**	**2/3/3/3/3**	**3/3/3/3/3**	**3/3/4/4/3**	**3/4/4/4/4**
	0.1093	0.4218	0.8437	3.6875	8.7343	22.7031
	0.1250	0.4062	2.5000	3.0937	13.4844	22.5313
	0.1250	0.3593	1.0156	3.1562	12.0938	23.2344
	0.1132	0.4531	2.0001	3.1875	10.1563	22.9688
	0.1520	0.3125	0.9687	2.8437	11.5000	31.2500
Necunoscute	48	102	234	261	432	896
n/m/p/q/g (secunde)	**4/4/4/4/4**	**4/4/5/5/4**	**4/5/5/5/5**	**5/5/5/5/5**	**5/5/6/6/5**	**6/6/6/6/6**
	31.4844	108.6090	226.3750	346.4690	783.063	2216.44
	34.7500	95.4063	208.5160	308.8910	816.953	2217.25
	38.2500	100.0160	211.8280	336.7750	873.813	2208.83
	48.0000	105.8440	236.0780	321.4250	802.891	2107.88
	38.2969	88.1075	205.8130	303.2660	848.469	2294.14
Necunoscute	864	1240	2225	2125	2850	4428

După cum se observă, algoritmul permite obținerea unor soluții pseudo optime în timp rezonabil în cazul rețelelor de transport de dimensiuni mari. Soluția furnizată de algoritm depinde de soluția admisibilă generată la *Pasul 1* de inițializare a algoritmului. De soluția inițială depinde și numărul de iterații necesar pentru soluționarea problemei neliniare și mai puțin de dimensiunea acesteia.

În Tabelul 3.6 sunt prezentate rezultatele executării mai multor probleme de transport pe rețea de diferite dimensiuni, pentru care se observă cum se modifică de la o iterație la alta mărimile $F0$ și $Z1$.

Tabelul 3.6. Modificarea valorilor F0 și Z1 pentru algoritmul AME2. Sursa: elaborat de autor

n/m/p/q/g / **Iterația**	**F0 și Z1** (u. c.)					
	3/3/3/3/3	**3/3/3/3/3**	**3/3/3/3/3**	**4/4/4/4/4**	**4/4/4/4/4**	**4/4/4/4/4**
1	49 / 41.96	67 / 51.39	58 / 42.90	74 / 54.79	65 / 61.66	62 / 46.42
2	41 / 34.67	46 / 44.67	39 / 35.36	50 / 50	57 / 55.70	44 / 39.36

3	29 / 29	35.5 / 31.33	35 / 33.78	46 / 44.66	47.5 / 45.8	39 / 39
4	-	29.66/ 29.66	31 / 31	44 / 42.76	41 / 41	37 / 37
5	-	-	-	44 / 44	39 / 39	-
6	-	-	-	44 / 44	-	-
	5/5/5/5/5	**5/5/5/5/5**	**5/5/5/5/5**	**6/6/6/6/6**	**6/6/6/6/6**	**6/6/6/6/6**
1	69 / 54.82	90 / 78	82 / 63.94	93 / 72.86	106 / 84.60	36 / 24.89
2	53 / 51.47	69 / 69	56.5 / 54.07	66.2 / 65.68	70.5 / 63.16	20.5 / 19.42
3	53 / 50.78	67 / 66.33	51 / 51	58 / 56.64	49 / 49	18 / 17.56
4	50 / 49.33	65 / 65	49.66 / 49	53 / 53	49 / 49	18 / 17.3
5	44.66 / 44.12	65 / 65	45 / 45	53 / 53	-	16 / 14.08
6	44 / 44	-	45 / 45	-	-	14 / 13.95
7	44 / 44	-	-	-	-	13.5 / 13.33
8	-	-	-	-	-	13 / 13
9	-	-	-	-	-	13 / 13

După cum se observă din datele prezentate în Tabelul 3.6, numărul de iterații până la îndeplinirea condiției de oprire a algoritmului nu depinde de dimensiunea problemei, ci doar de soluția inițială de la *Pasul 1*. De la o iterație la alta, valoarea $F0$ este mai mică sau egală decât valoarea $F0$ obținută la iterația precedentă. Cazul când $F0$ rămâne nemodificat pe parcursul a câtorva iterații demonstrează că problemele de acest tip pot avea mai multe minime locale. Pe de altă parte, aceste puncte permit diferite aproximări ale funcției neliniare cu una liniară, ceea ce duce la o ieșire din blocaj și la obținerea unui alt minim local cu valoarea funcției de cost mai mică. În cazul când $F0$ și $Z1$ coincid dar algoritmul nu se stopează spune despre faptul că aceste valori sunt obținute în puncte diferite.

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

3.2.4 Algoritmul genetic AMG2

În cele ce urmează este soluționată problema *PNTR3I*, aplicând un algoritm genetic, ceea ce presupune codificarea problemei astfel încât fiecărui cromozom din populație să i se poată asocia doar o soluție admisibilă. Fiecare populație nou-creată păstrează cromozomii cu cele mai bune caracteristici din populația precedentă, astfel încât în timpul selecției să nu fie pierdute soluțiile cărora le sunt asociate costuri mici.

Fiecare operator de încrucișare și mutație este descris astfel încât ori de câte ori sunt aplicați asupra cromozomilor care descriu populația, există o singură soluție admisibilă care i se asociază după decodificare. La baza codificării soluțiilor admisibile a problemei este folosită noțiunea de arbore de acoperire care descrie corect o soluție a problemei. Un element important în descrierea algoritmilor genetici este și dimensiunea populațiilor generate care trebuie să fie reprezentativă și să permită determinarea unei soluții în timp rezonabil chiar și pentru probleme de dimensiuni mari. În algoritmul AMG2 descris mai jos, dimensiunea populației este de $4|V|$ cromozomi, astfel este studiat un număr suficient de soluții admisibile.

Deoarece fiecare cromozom trebuie să conțină informație despre fiecare indice care descrie problema, el este divizat în 5 compartimente: *primul* conține n arbori de acoperire fiecare cu rădăcina în una din sursele rețelei de transport descrișI de arce, *al doilea* conține o permutare a numerelor $i = \overline{1,n}$ asociate surselor și descrie ordinea lor de deservire, *al treilea* conține o permutare a numerelor $j = \overline{1,m}$ asociate destinațiilor și descrie ordinea lor de deservire, *al patrulea* conține o permutare a numerelor $k = \overline{1,p}$ asociate tipurilor de produse și descrie în ce ordine ele sunt transportate, *al cincilea* conține o permutare a numerelor $l = \overline{1,q}$ asociate tipurilor de transport și descrie în ce ordine ele pot fi utilizare.

Algoritmul genetic AMG2 propus pentru soluționarea *PNTR3I* cu funcții concave nedescrescătoare secvențial-liniare constă din următorii pași:

Pasul 1 *Inițializarea.* Se generează aleatoriu populația inițială din $4|V|$ cromozomi fiecare descris de mulțimea $P = \left\{\{T_r, r = \overline{1,n}\}, P_n, P_m, P_p, P_q\right\}$, unde $T_r = \{(i,j) \mid i,j = \overline{1,n+m+g}\}$ – arbori de acoperire cu rădăcina în sursa r, $r = \overline{1,n}$ a rețelei de transport, $P_n = \{p(1), p(2), \dots, p(n)\}$ – o permutare a numerelor $i = \overline{1,n}$ asociate surselor, $P_m = \{p(1), p(2), \dots, p(m)\}$ – o permutare a numerelor $j = \overline{1,m}$ asociate destinațiilor, $P_p = \{p(1), p(2), \dots, p(p)\}$ – o permutare a numerelor $k = \overline{1,p}$ asociate tipurilor de produse, $P_q = \{p(1), p(2), \dots, p(q)\}$ – o permutare a numerelor $l = \overline{1,q}$ asociate tipurilor de transport.

Pasul 2 *Decodificarea și Evaluarea cromozomilor.* Decodificarea presupune că fiecărui cromozom i se asociază o soluție admisibilă de forma $x = (x_{111}, x_{112}, \dots, x_{epq})$ transportând fluxul din k produse cu l tipuri de transport din fiecare sursă către destinații pe arcele ce corespund arborilor. În acest scop:

1. Se creează tablourile unidimensionale care conțin respectiv cantitățile de produs în surse, necesitățile de produs în destinații, tipurile de produce cu cantități diferite și tipurile de transport cu capacități diferite astfel:

$\alpha' = [\alpha'_1, \alpha'_2, \dots, \alpha'_n], \beta' = [\beta'_1, \beta'_2, \dots, \beta'_m], \gamma' = [\gamma'_1, \gamma'_2, \dots, \gamma'_p], \delta' = [\delta'_1, \delta'_2, \dots, \delta'_q]$.

2. Pentru fiecare indice $r \in P_{ij}$ se determină soluția intermediară astfel:

a) pentru fiecare indice $j \in P_m$, fiecare indice $k \in P_p$ și fiecare indice $l \in P_q$ se determină: $x_{ekl} = min\{\alpha'_i,\ \beta'_j,\ \gamma'_k,\ \delta'_l\}$, după care are loc actualizarea $\alpha'_i := \alpha'_i - x_{ekl}$, $\beta'_j := \beta'_j - x_{ekl}$, $\gamma'_k := \gamma'_k - x_{ekl}$, $\delta'_l := \delta'_l - x_{ekl}$, unde e - indicele arcului $(i,j) \in T_r$;

b) arcului (i,j) cu indicele e, unde $j \notin V_t$, i se asociază fluxul:

$$x_{rekl} = \begin{cases} 0, & (i,j) \notin T_r, \\ \sum\limits_{(j,j') \in T_r} x_{re'kl} & e' - indicele\ arcului\ (j,j'), \end{cases}$$

unde $k = \overline{1,p}$, $l = \overline{1,q}$.

Ca rezultat se obțin soluțiile intermediare de forma:

$$x_{rekl} = (x_{r111}, x_{r112}, \dots, x_{rupq}), r = \overline{1,n}$$

iar soluția finală a problemei are forma:

$$x = \left(\sum_{i=1}^{n} x_{i111}, \sum_{i=1}^{n} x_{i112}, \dots, \sum_{i=1}^{n} x_{iupq} \right).$$

Evaluarea cromozomilor presupune determinarea valorii funcției obiectiv pentru fiecare dintre soluțiile asociate cromozomilor.

Pasul 3 *Selectarea cromozomilor.* Cromozomii sunt sortați în ordinea creșterii valorii funcției obiectiv pentru soluția asociată cromozomului. Cromozomii din prima jumătate a populației sunt transferați în noua populație.

Pasul 4 *Încrucișarea cromozomilor.* Încrucișarea se realizează între cromozomii transferați în populația $P(i)$ din populația $P(i-1)$ prin selecție. Aleatoriu este aplicată aceeași tăietură ambilor cromozomi-părinți asupra compartimentului 1. Cromozomii-urmași se obțin astfel:

- primul cromozom-urmaș primește arborii de până la tăietură a cromozomului-mamă cu ordinea respectivă a surselor și a tipurilor de produse și arborii de după tăietura a cromozomul-tată cu ordinea respectivă a destinațiilor și a tipurilor de transport;
- al doilea cromozom-urmaș primește arborii de până la tăietură a cromozomului-tată cu ordinea respectivă a surselor și tipurilor de produse și arborii de după tăietură a cromozomul-mamă cu ordinea respectivă a destinațiilor și tipurilor de transport.

Fiecare pereche de cromozomi-părinți generează doi cromozomi-urmași și dimensiunea populației noi rămâne constantă.

Pasul 5 *Mutația.* Unei gene a cromozomului-urmaș i se aplică o mutație cu probabilitatea $\varepsilon \in [0.1,0.5]$ și presupune:

- pentru fiecare compartiment *doi* – surse, *trei* – destinații, *patru* – tipuri de produse și *cinci* – tipuri de transport, aleatoriu se decide dacă este aplicată mutația;
- în fiecare dintre compartimentele alese se schimbă valorile între două elemente alese aleatoriu.

Pasul 6 *Verificarea condiției de oprire.* Implică oprirea algoritmului după k iterații. În calitate de soluție a problemei servește soluția care corespunde cromozomului cu valoarea funcției obiectiv minimă din ultima populație construită. Se trece la *Pasul 2* dacă condiția de oprire nu este satisfăcută.

Remarca 3.4. *În cazul problemelor de dimensiuni mari, drept condiție de oprire a algoritmului poate servi durata de execuție în timp. Deci, după o perioadă anumită de timp este stopată generarea noilor populații, iar ca soluție a problemei servește soluția care corespunde cromozomului cu cele mai bune caracteristici din ultima populație.*

Remarca 3.5. *Algoritmul AMG2 este aplicat în cazul rețelelor care îndeplinesc următoarele condiții:*

1. *Graful rețelei de transport este conex și aciclic;*
2. *Există cel puțin 2 surse, 2 destinații, 2 tipuri de produse, 2 tipuri de transport și cel puțin un punct intermediar;*
3. *Există cel puțin un drum din fiecare sursă în fiecare destinație;*
4. *Vârfurile surse nu conțin arce ascendente, iar vârfurile destinații nu conțin arce descendente.*

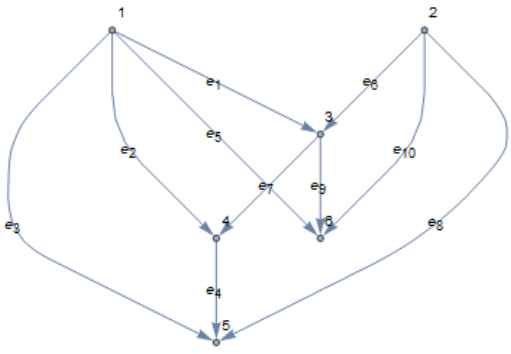


Fig. 3.4. Rețea cu două surse și două destinații.
Sursa: elaborat de autor

Exemplul (codificare) 3.5. Se consideră rețeaua de transport, descrisă de graful din Figura 3.4, cu două puncte intermediare, sursele A_1 și A_2 care conțin respectiv $\alpha_1 = 20$ u. c. și $\alpha_2 = 80$ u. c. de produs de tipul P_1 și P_2 respectiv câte $\gamma_1 = 50$ u. c. și $\gamma_2 = 50$ u. c. fiecare. Produsele sunt transportate cu ajutorul mijloacelor de transport T_1 și T_2 de capacitatea respectivă de $\delta_1 = 60$ u. c. și $\delta_2 = 40$ u. c. în destinațiile B_1 și B_2 care au necesitățile respective de $\beta_1 = 70$ u. c. și $\beta_2 = 30$ u. c.

Un cromozom a populației, în acest caz este descris de:

- *compartimentul 1* care conține doi arbori de acoperire $T_1 = \{(1,3), (1,4), (4,5), (1,6)\}$ și $T_2 = \{(2,3), (3,4), (2,5), (3,6)\}$ cu rădăcina în sursa 1 și 2 respectiv;
- *compartimentul 2* care conține ordinea de deservire a surselor, fie {1,2};
- *compartimentul 3* care conține ordinea de deservire a destinațiilor, fie {2,1} ceea ce înseamnă că este aprovizionat vârful 6 (a doua destinație) apoi vârful 5 (prima destinație);
- *compartimentul 4* care conține ordinea de deservire a diferitor tipuri de produse, fie {1,2};
- *compartimentul 5* care conține ordinea de utilizare a diferitor tipuri de transport, fie {2,1}

Astfel, se obține un cromozom de forma:

{{(1,3), (1,4), (4,5), (1,6)},{(2,3), (3,4), (2,5), (3,6)}},{1,2},{2,1},{1,2},{2,1}}.

Exemplul (decodificare) 3.6. Pentru cromozomul generat se construiește soluția admisibilă de forma $x = (x_{111}, x_{112}, x_{121}, x_{122}, x_{211}, x_{212}, x_{221}, x_{222}, x_{311}, x_{312}, x_{321}, x_{322}, x_{411}, x_{412}, x_{421}, x_{422}, x_{511}, x_{512}, x_{521}, x_{522}, x_{611}, x_{612}, x_{621}, x_{622}, x_{711}, x_{712}, x_{721}, x_{722}, x_{811}, x_{812}, x_{821}, x_{822}, x_{911}, x_{912}, x_{921}, x_{922}, x_{10\,11}, x_{10\,12}, x_{10\,21}, x_{10\,22})$ astfel:

Se creează tablourile $\alpha' = [20, 80]$, $\beta' = [70,30]$, $\gamma' = [50,50]$, $\delta' = [60,40]$.

- Este transportat produs prin arborele de acoperire cu rădăcina în sursa 1:

Din sursa 1 este transportat produs în destinația a doua (nodul 6), deci $x_{512} = min\{20,30,50,40\} = 20$, $\alpha_1' := 20 - 20 = 0$, $\beta_2' := 30 - 20 = 10$, $\gamma_1' := 50 - 20 = 30$, $\delta_2' := 40 - 20 = 20$. Deoarece este transportat tot produsul din sursă, se asociază tuturor arcelor rămase, din acest arbore, un flux nul.

- Este transportat produs prin arborele de acoperire cu rădăcina în sursa 2:

În destinația a doua prin arcul 9 este transportat fluxul $x_{912} = min\{80,10,30,20\} = 10$, $\alpha_2' := 80 - 10 = 70$, $\beta_2' := 10 - 10 = 0$, $\gamma_1' := 30 - 10 = 20$, $\delta_2' := 20 - 10 = 10$. În prima destinație (nodul 5) prin arcul 4 este transportat fluxul $x_{412} = min\{70,70,20,10\} = 10$, $\alpha_2' := 70 - 10 = 60$, $\beta_1' := 70 - 10 = 60$, $\gamma_1' := 20 - 10 = 10$, $\delta_2' := 10 - 10 = 0$, fluxul $x_{411} = min\{60,60,10,60\} = 10$, $\alpha_2' := 60 - 10 = 50$, $\beta_1' := 60 - 10 = 50$, $\gamma_1' := 10 - 10 = 0$, $\delta_1' := 60 - 10 = 50$, fluxul $x_{421} = min\{50,50,50,50\} = 50$, $\alpha_2' := 50 - 50 = 0$, $\beta_1' := 50 - 50 = 0$, $\gamma_2' := 50 - 50 = 0$, $\delta_1' := 50 - 50 = 0$.

Parcurgând arborele de acoperire în adâncime, fluxul este repartizat astfel: fluxul de pe arcul 4 este asociat și arcului 7 astfel: $x_{711} = 10$, $x_{712} = 10$, $x_{721} = 50$. Arcului 6 îi este asociat fluxul total de pe arcele 7 și 9 astfel: fluxul $x_{611} = x_{711} + x_{911} = 10 + 0 = 10$, fluxul $x_{612} = x_{712} + x_{912} = 10 + 10 = 20$, fluxul $x_{621} = x_{721} + x_{921} = 50 + 0 = 50$, fluxul $x_{622} = x_{722} + x_{922} = 0 + 0 = 0$. Pentru restul arcelor, prin care nu este transportat flux, este asociat un flux nul.

Ca rezultat, se obține soluția $x = (0,0,0,0,0,0,0,0,0,0,0,0,10,10,50,0,0,20,0,0,10,20,50,0,$ $10, 10,50,0,0,0,0,0,0,10,0,0,0,0,0,0)$. Evaluarea presupune determinarea valorii funcției obiectiv în această soluție.

Exemplul (încrucișare) 3.7. Pentru rețeaua din Figura 3.4, se consideră doi cromozomi-părinți care participă la încrucișare. Tăietura este aplicată la mijlocul mulțimii de arbori din compartimentul 1. Pentru:

Cromozom-mama {{(1,3),(1,4),(4,5),(1,6)},{(2,3),(3,4),(2,5),(3,6)}},{1,2},{2,1},{2,1},{2,1}}

Cromozom-tata {{(1,3),(3,4),(1,5),(1,6)},{(2,3),(3,4),(4,5),(2,6)}},{1,2},{2,1},{1,2},{1,2}}

cromozomii-urmaşi sunt:

Cromozom-urmaş1 {{(1,3),(1,4),(4,5),(1,6)},{ (2,3),(3,4),(4,5),(2,6)}},{1,2},{2,1},{2,1},{1,2}}

Cromozom-urmaş2 {{(1,3),(3,4),(1,5),(1,6)},{ (2,3),(3,4),(2,5),(3,6)}},{1,2},{2,1},{1,2},{2,1}}

Exemplul (mutație) 3.8. În exemplul prezentat mai jos sunt selectate mulțimile care descriu ordinea de deservire a destinațiilor și a tipurilor de transport pentru care sunt schimbate cu locul două elemente din fiecare mulțime în parte, după cum urmează:

Cromozom-urmaş {{(1,3),(1,4),(4,5),(1,6)},{ (2,3),(3,4),(4,5),(2,6)}},{1,2},{2,1},{2,1},{1,2}}

după mutație cromozomul modificat este:

Cromozom-urmaş {{(1,3),(1,4),(4,5),(1,6)},{ (2,3),(3,4),(4,5),(2,6)}},{1,2},{1,2},{2,1},{2,1}}.

Se formulează şi se demonstrează următoarele teoreme:

Teorema 3.8. *Algoritmul AMG2 necesită un volum de memorie de ordinul* $O(|V|(n|V| + n + m + p + q))$.

Demonstrație. *Datele de intrare descrise de tabelul de restricții este de dimensiunea* upq. *Pentru păstrarea soluției asociate cromozomului cu cele mai bune caracteristici din populație este necesară* upq *memorie. Păstrarea unui cromozom necesită* $n|V| + n + m + p + q$ *memorie, iar pentru întreaga populație –* $4|V|(n|V| + n + m + p + q)$ *memorie. Prin urmare, algoritmul AMG2 necesită un volum de memorie de ordinul* $O(|V|(n|V| + n + m + p + q))$.□

Teorema 3.9. *Complexitatea unei iterații a algoritmului AMG2 este de ordinul* $O(npq|V|(m + |V| + u))$.

Demonstrație. *Pentru inițializarea datelor de intrare descrise de tabelul de restricții sunt necesare* $O(upq)$ *operații. Generarea unui cromozom necesită* $O(n|V| + n + m + p + q)$

operații, astfel încât generarea întregii populații necesită $O(|V|(n|V| + n + m + p + q))$ *operații. Evaluarea soluției asociate unui cromozom necesită* $O(n(mpq + |V|pq + upq))$ *operații. Deoarece în populație sunt* $4|V|$ *cromozomi, complexitatea de evaluare a tuturor soluțiilor asociate cromozomilor este* $O(npq|V|(m + |V| + u))$. *Încrucișarea este de o complexitate* $O(n|V| + n + m + p + q)$ *pentru un cromozom și de o complexitate* $O(|V|(n|V| + n + m + p + q))$ *pentru întreaga populație. Prin urmare, o iterație a algoritmului AMG2 are complexitatea de ordinul* $O(n|V| + n + m + p + q)$.□

Observația 3.3. *Din Teorema 3.9 rezultă că timpul de execuție a algoritmului AMG2 este* $O(U(n|V| + n + m + p + q))$, *unde* U *este numărul de iterații necesar pentru obținerea unei soluții asociate cromozomului cu caracteristici bune.*

Teorema 3.10. *Algoritmul AMG2 converge către optimul local.*

Demonstrație. *Din populația* $P(\mathrm{i}-1)$ *sunt transferați în populația* $P(i)$ *cromozomii pentru care valoarea funcției obiectiv în soluția asociată este minimă. Cromozomilor populațiilor construite la fiecare iterație le este asociată o soluție admisibilă, deci se poate spune că după executarea unui număr finit de iterații este depistat cromozomul căruia îi este asociată soluția ce descrie optimul local. Prin urmare, algoritmul AMG2 converge către optimul local.*□

3.2.5 Implementarea și testarea algoritmului AMG2

Algoritmul AMG2 este implementat în limbajul Wolfram și testat în baza unui șir de exemple, probleme de transport de diferite dimensiuni ca număr de surse, număr de destinații, număr de puncte intermediare, număr de tipuri diferit de produse și număr de tipuri diferite de transport care pot fi utilizate.

Un exemplu practic de aplicare a algoritmului AMG2 pentru soluționarea problemei neliniare de transport pe rețea descrisă de surse, destinați, puncte intermediare, tipuri de produse și tipuri de transport este prezentat în Anexa 12. Codul scris în Wolfram Mathematica poate fi accesat pe https://github.com/TatianaPasa/GeneticAlgorithm.

În Tabelele 3.7 și 3.8 sunt prezentate rezultatele soluționării unui șir de probleme de diferite dimensiuni, unde funcțiile de cost sunt funcții concave secvențial-liniare. Dimensiunea problemei de transport este descrisă de n – surse, m – destinații, p – tipuri de produse transportate, q – tipuri de transport utilizate și g – puncte intermediare. În baza funcției de producere și consum totalul produselor disponibile în surse este egal cu totalul produselor necesar în destinații și este egal cu 50 u. c. Totalul de produse de diferite tipuri coincide cu capacitatea totală de tipuri de transport utilizat și este de 50 u. c.

Tabelul 3.7. Timpul de execuție pentru algoritmul AMG2. Sursa: elaborat de autor

n/m/p/q/g (secunde)	**2/2/2/2/2**	**2/3/2/3/2**	**2/3/3/3/3**	**3/3/3/3/3**	**3/4/3/4/3**	**3/4/4/4/4**	**4/4/4/4/4**	**4/5/4/5/4**
1	0.2500	0.5156	0.9687	1.3125	2.0312	3.9062	4.3750	6.6406
2	0.2656	0.4218	0.9375	1.2500	2.0937	3.9531	4.3437	6.5781
3	0.2812	0.4375	0.9687	1.2500	2.0312	3.9531	4.5468	6.6562
4	0.3437	0.5312	1.0000	1.2968	2.0000	3.9218	4.4531	6.5937
5	0.3281	0.4843	1.0312	1.2500	2.0625	3.9062	4.4531	6.5468
Necunoscute	48	102	234	261	432	896	864	1240
n/m/p/q/g (secunde)	**4/5/5/5/5**	**5/5/5/5/5**	**5/6/5/6/5**	**6/6/6/6/6**	**7/7/7/7/7**	**8/8/8/8/8**	**9/9/9/9/9**	**10/10/10/10/10**
	11.6250	12.2656	17.5313	29.6719	62.1563	118.938	213.656	353.234
	11.6406	12.4375	17.5469	29.5156	62.0000	117.906	211.906	359.266
	11.5156	12.5781	17.5938	29.4531	62.7344	118.922	216.281	361.453
	11.5313	12.4063	17.5625	29.6406	62.0625	117.375	211.656	356.750
	11.4844	12.4844	17.6406	20.9531	62.6406	1117.031	216.406	355.703
Necunoscute	2225	2125	2850	4428	8232	14080	22599	34500

Testele sunt efectuate pe o mașină Intel i5-2500 cu 4 Cores și 8 GB memorie DDR3 în Wolfram Mathematica 12.

În Tabelul 3.7 este dat timpul de execuție a algoritmului AMG2 pentru probleme de diferite dimensiuni: numărul de surse, destinații, puncte intermediare, tipuri de produse și tipuri de transport, pentru a căror soluționare sunt utilizate de la 48 de necunoscute în cazul problemei cu 2 surse, 2 destinații, 2 tipuri de produse, 2 tipuri de transport și 2 puncte intermediare până la 34500 de necunoscute în cazul problemei cu 10 surse, 10 destinații, 10 tipuri de produse, 10 tipuri de transport și 10 puncte intermediare. După cum se poate observa, algoritmul permite obținerea unor soluții pseudo optime în timp rezonabil chiar și în cazul problemelor de transport de dimensiuni mari, timpul fiind dat după generarea a 10 populații.

În Tabelul 3.8 sunt expuse datele ce vizează rezultatele soluționării unui șir de probleme de transport pe rețea de dimensiuni diferite, unde se poate observa cum pentru fiecare din cele 10 populații generate se modifică valoarea funcției obiectiv calculată pentru soluția asociată cromozomului cu cele mai bune caracteristici din populația respectivă și valoarea funcției obiectiv totală pentru populația respectivă.

Din datele prezentate în Tabelul 3.8 se observă că $F_T(x)$ în general descrește de la o iterație la alta, cu unele excepții – când în urma operațiilor de încrucișare și mutație se obțin soluții cărora

le corespund costuri mai ridicate decât cele ce corespundeau populației precedente. Mărirea nesemnificativă a $F_T(x)$ este urmată de descreșteri, ceea ce sugerează că în populațiile noi se conțin preponderent cromozomi cu caracteristici mai bune decât cei din populația obținută la iterația anterioară.

Tabelul 3.8. Modificarea valorii $F(x)$ și $F_T(x)$ pentru algoritmul AMG2.
Sursa: elaborat de autor

Iterația	**n/m/p/q/g** (u. c.)				
	4/4/4/4/4	**5/5/5/5/5**	**6/6/6/6/6**	**7/7/7/7/7**	**10/10/10/10/10**
I	30 / 2431	43 / 4078	60 / 6301	73 / 8897	94 / 17495
II	30 / 2263	43 / 3767	51 / 5837	73 / 8560	94 / 17176
III	29 / 2113	43 / 3584	51 / 5691	69 / 8059	94 / 16846
IV	29 / 2117	42 / 3343	51 / 5317	61 / 7774	94 / 16546
V	28 / 1885	41 / 3208	50 / 5145	61 / 7657	88 / 16157
VI	28 / 1890	40 / 3033	45 / 4878	55 / 7529	88 / 15799
VII	26 / 1851	37 / 2923	45 / 4744	55 / 7219	88 / 15839
VII	17 / 1831	32 / 2801	45 / 4619	55 / 7286	85 / 15480
IX	17 / 1826	32 / 2775	44 / 4559	55 / 7126	85 / 15403
X	17 / 1819	32 / 2774	40 / 4551	55 / 6865	85 / 15299

Deși se poate vedea că la câteva iterații consecutive se repetă aceeași valoare $F(x)$, ceea ce înseamnă că la câteva iterații consecutive nu este depistat un cromozom cu caracteristici mai bune, se atrage atenția că algoritmul permite ieșirea dintr-un astfel de blocaj determinând un nou cromozom cu caracteristici mai bune deci și o soluție pentru care valoarea funcției obiectiv este mai mică.

Algoritmul permite determinarea unei soluții optime locale pentru care valoarea funcției obiectiv este cu atât mai mică cu cât sunt îndeplinite mai multe iterații ale algoritmului. Numărul de iterații depinde de dimensiunea problemei soluționate, de timpul disponibil pentru a furniza soluția dar și de capacitățile tehnologice de care se dispune.

Dacă se soluționează aceleași probleme neliniare de transport cu 3 indici aplicând de câteva ori algoritmii AME2 și AMG2, se poate concluziona că rezultatul furnizat depinde de pasul de inițializare. Deși algoritmul AMG2 depinde de populația inițială, după un număr suficient de iterații el permite obținerea soluției optime locale din vecinătatea optimului global, deci a unei soluții pseudo optime. În cazul algoritmului AME2 doar o alegere corectă a soluției inițiale permite obținerea soluției optime locale din vecinătatea optimului global, fapt ce nu poate fi realizat mai

ales în cazul problemelor de dimensiuni mari. O caracteristică importantă a lui AMG2 este ieșirea cu succes din blocaje în minime locale și, astfel, îmbunătățirea soluției.

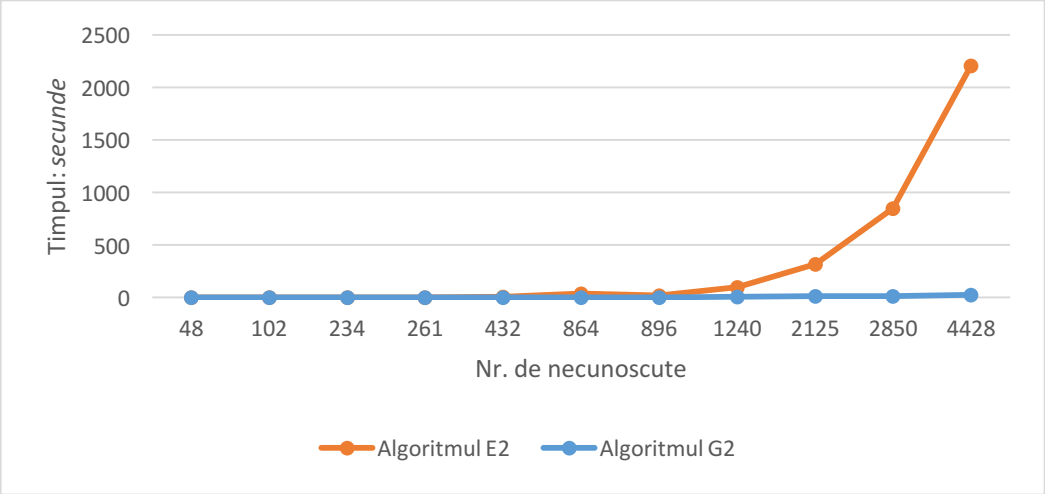


Fig. 3.5. Timpul de execuție pentru AME2 și AMG2. Sursa: elaborat de autor

Din Figura 3.5 se poate observa că algoritmul AMG2 furnizează soluție optimă locală, după 10 iterații, în timp de cca un minut chiar și pentru probleme cu peste 1500 de necunoscute, iar în cazul aplicării algoritmului AME2, când numărul necunoscutelor trece de o mie, timpul de furnizare a soluției crește semnificativ.

3.3 Concluzii la capitolul 3

În capitolul 3 sunt formulate problemele neliniare de transport cu 4 și cu 5 indici în care circulă câteva tipuri de produse transportate de câteva tipuri de transport. Pentru problemele formulate sunt propuși algoritmi care permit obținerea soluției în timp rezonabil și pentru probleme de dimensiuni mari. În legătură cu studierea problemelor neliniare de transport cu mai mulți indici au fost obținute rezultate care completează rezultatele cunoscute la moment în acest domeniu și se referă la:

1) elaborarea algoritmului AME1 de soluționare a problemei neliniare de transport cu 4 indici, descrisă de funcții concave de cost, care presupune reducerea consecutivă a problemei neliniare la o problemă liniară;
2) elaborarea algoritmului AMG1 de soluționare a problemei neliniare de transport cu 4 indici, descrisă de funcții concave de cost, care are la bază aplicarea operatorilor de selectare, încrucișare și mutație a algoritmilor genetici;
3) formularea problemei neliniare de transport pe rețea cu 5 indici descrisă de surse, destinații, puncte intermediare, tipuri de produs și tipuri de transport ca problemă

neliniară de transport pe rețea cu 3 indici descrisă de produse, tipuri de transport și mulțimea de arce;

4) elaborarea algoritmului AME2 de soluționare a problemei neliniare de transport pe rețea cu 3 indici, descrisă de funcții concave de cost, care presupune reducerea consecutivă a problemei neliniare la o problemă liniară;
5) elaborarea algoritmului AMG2 de soluționare a problemei neliniare de transport pe rețea cu 5 indici formulată ca problemă cu 3 indici, descrisă de funcții concave de cost, care are la bază aplicarea operatorilor de selectare, încrucișare și mutație a algoritmilor genetici;
6) descrierea unei codificări corecte a problemei respective, pentru fiecare dintre algoritmii genetici AMG1 și AMG2, astfel încât la decodificare se obține o singură soluție admisibilă pentru fiecare cromozom;
7) descrierea operatorilor de încrucișare și mutație, astfel încât ori de câte ori sunt aplicați, în cadrul fiecărui dintre algoritmii genetici AMG1 și AMG2, după decodificarea cromozomului nou construit se obține o soluție admisibilă a problemei codificate;
8) demonstrarea convergenței algoritmilor AME1, AME2, AMG1 și AMG2 către un optim local;
9) determinarea volumului necesar de memorie pentru implementarea practică a algoritmilor elaborați;
10) testarea algoritmilor implementați în Sistemul Mathematica care permit soluționarea problemelor neliniare de transport.

Potrivit studiului efectuat în acest capitol se formulează următoarele concluzii:

1. Algoritmul AME1 permite soluționarea problemei neliniare de transport cu 4 indici, prin reducerea acesteia la o problemă liniară, în timp rezonabil pentru probleme de până la 1000 de necunoscute.
2. Algoritmul AME2 permite soluționarea problemei neliniare de transport pe rețea cu 5 indici formulată ca problemă cu 3 indici, prin reducerea acesteia la o problemă liniară, în timp rezonabil pentru probleme de până la 1000 de necunoscute.
3. Algoritmii genetici AMG1 și AMG2 propuși pentru soluționarea problemelor neliniare de transport formulate reduc esențial timpul de obținere a soluției, astfel încât pot fi rezolvate probleme cu câteva mii de necunoscute în timp rezonabil.
4. Complexitatea unei iterații pentru fiecare dintre algoritmii genetici AMG1 și AMG2 este una polinomială.

5. Fiecare dintre algoritmii AMG1 și AMG2 generează o soluție pseudo optimă care este o soluție locală din vecinătatea soluției optime globale în cazul generării unui număr suficient de populații.
6. Pentru algoritmii AMG1 și AMG2 este demonstrat experimental că permit ieșirea din blocaje în soluții locale datorită operațiilor de încrucișare și mutație.

CONCLUZII GENERALE ȘI RECOMANDĂRI

Cercetările efectuate în cadrul tezei "***Algoritmi de soluționare a problemelor neliniare de transport***" corespund scopului și obiectivelor expuse în introducerea lucrării. Ținând cont de importanța și aplicabilitatea problemei studiate în capitolul 1 și în baza rezultatelor obținute în capitolele 2 și 3, se formulează câteva concluzii generale și recomandări.

Concluzii generale asupra rezultatelor obținute

1. Soluționarea problemelor neliniare de transport de dimensiuni mari descrise de rețele cu una sau mai multe surse și una sau mai multe destinații aplicând algoritmii euristici AE1 și AE2, descriși de autor, permit îmbunătățirea esențială a timpului de execuție la aflarea soluției pseudo optime de minimum 600 de ori pentru probleme de dimensiuni mici și de peste 9 000 000 de ori pentru probleme de dimensiuni mari (Cap. 2, 2.2);
2. Algoritmii genetici AG1, AG2, AG3 și AG4, propuși de autor, permit soluționarea în timp rezonabil a problemelor neliniare de transport cu câteva mii de necunoscute descrise de rețelele cu una sau mai multe surse și una sau mai multe destinații pentru care funcțiile standard nu sunt aplicabile în timp real (Capitolul 2, 2.3, 2.4, 2.5);
3. Soluționarea problemelor neliniare de transport de dimensiuni mari descrise de rețele cu mai mulți indici aplicând algoritmii euristici AME1 și AME2, descriși de autor, permit îmbunătățirea esențială a timpului de execuție și generează o soluție pseudo optimă în timp rezonabil (Capitolul 3, 3.1.2, 3.1.3, 3.2.2, 3.2.3);
4. Algoritmii genetici AMG1 și AMG2, propuși de autor, permit soluționarea în timp rezonabil a problemelor neliniare de transport cu câteva mii de necunoscute, descrise de rețelele cu mai mulți indici (Capitolul 3, 3.1.4, 3.1.5, 3.2.4, 3.2.5);
5. Testarea algoritmilor genetici AG3, AG4, AMG1 și AMG2, demonstrează că se asigură cu succes ieșirea din blocajele în soluțiile locale, iar procedura de codificare permite la decodificare să se obțină întotdeauna soluții admisibile ale problemei examinate (Capitolul 2, 2.4.3, 2.5.3, Capitolul 3, 3.1.5, 3.2.5);
6. Se observă o îmbunătățire a timpului de execuție pentru algoritmii genetici AG1, AG2, AG3, AG4, AMG1, AMG2 în comparație cu timpul de execuție a algoritmilor AE1, AE2, AME1 și AME2, primii fiind aplicabili și pentru probleme de dimensiuni foarte mari de ordinul a zeci de mii de necunoscute în timp de la 5 la 10 minute pe calculatoarele menționate în teză (Capitolul 2, 3).

Algoritmii elaborați și testați în cadrul acestei lucrării sunt realizați în limbajul Wolfram și implementați în Sistemul Mathematica.

Avantajele şi valoarea elaborărilor propuse

Cercetările efectuate sunt o extindere a teoriei ce ține de problemele neliniare de transport pe rețea cu funcții concave de cost. Datorită gradului înalt de noutate și originalitate, algoritmii elaborați propuși în lucrare au o valoare științifică distinctă. Rezultatele obținute pot fi utilizate în diverse domenii atât teoretice, cât și practice.

Recomandări

Luând în considerare importanța teoretico-aplicativă și complexitatea problemelor abordate în lucrare, se recomandă:

1. Continuarea cercetărilor în vederea elaborării unor noi algoritmi care permit obținerea soluțiilor optime pentru probleme neliniare de transport cu variate tipuri de funcții de cost, diferite de cele examinate deja în lucrare.
2. Extinderea rezultatelor obținute pentru cazuri nestudiate ale problemelor neliniare de transport cu mai mulți indici, prin descrierea modelului matematic și a algoritmilor care permit soluționarea lor.
3. Utilizarea rezultatelor obținute în curricula unor discipline opționale predate în cadrul instituțiilor de învățământ superior la diverse cicluri de studii universitare.

BIBLIOGRAFIE

1. ABDI, S., BAROUGHI, F., ALIZADEH, B. The Minimum Cost Flow Problem of Uncertain Random Network. In: *Asia-Pacific Journal of Operational Research , 2018, vol. 35*, no. 03. Print ISSN 0217-5959, Online ISSN 1793-7019.
2. AHMED, M. M., TANVIR, A. M., SULTANA, S., MAHMUD, S., UDDIN, M. S. *Modification to solve transportation problems. A cost minimization approach.* Preluat în ianuarie 2020, In: Annals of Pure and Applied Mathematics, 2014, Disponibil: https://www.researchgate.net/publication/273631960_An_Effective_Modification_to_Solve_Transportation_Problems_A_Cost_Minimization_Approach.
3. AHMED, M. M., KHAN, A. R., AHMED, F., UDDIN, M. S. Incessant Allocation Method for Solving Transportation Problems. In: *American Journal of Operations Research, 2016, vol. 6*, p. 236-144. ISSN 2160-8830.
4. AHUJA, R. K., GOLDBERG, A. V., ORLIN, J. B., TARJAN, R. E. Finding minimum-cost flows by double scaling. In: *Sloan W. P., 1988, no. 2047-88*. [citat august 2018]. Disponibil: https://dspace.mit.edu/bitstream/handle/1721.1/47961/findingminimumco00sloa.pdf%3Bjsessionid%3DF0ABD3316290E7610A2E42EF46C5A08E?sequence%3D.
5. ARMSTRONG, R. D., JIN, Z. A new strongly polinomial dual network simplex algoritm. In: *Math. Program., 1997, vol. 78*, p. 131-148. https://doi.org/10.1007/BF02614366.
6. BAKHAYT, A.-G. K. Solving problem by using PSO. In: *Sci. Int., 2016, vol. 28,* no. 3, p. 2403-2410.
7. BAO, D., Gu, J., DI, Z., ZHANG, T. Optimization of Airport Shuttle Bus Routes Based on Travel Time Reliability. In: *Hindawi, Mathematical Problems in Engineering, Article, 2018,* ID 2369350, https://doi.org/10.1155/2018/2369350, 12 pages.
8. BAUMSTARK, N., BLELLOCH, G., SHUN, I. Efficient Implementation of Syncronouns Parallel Push-Relabel Algorithm, 2015. [citat decembrie 2017]. Disponibil: https://arxiv.org/pdf/1507.01926.pdf.
9. BLAND, R. G., JENSEN, D. L. On the computation behavir of a polynomial-time network flow algorithm. In: *Technical Report 661, School of Operations Research and Industrial Engineering*, 1985.
10. BOHEME, T. J., FRANK, B. *Hybrid Systems, Optimal Control and Hybrid Vehicles.* Springer International Publishing AG, 2017. DOI: 10.1007/978-3-319-51317-1.

11. BORRADAILE, G., KLEIN, P. An O(nlogn) algorithm for maximum st-flow in a directed planar graph. In: *Journal of the ACM, 2009, vol. 56,* no. 2, p. 1-34. Disponibil: https://dl.acm.org/doi/pdf/10.1145/1502793.1502798.
12. BORRADAILE, G., KLEIN, P., MOZES, S., NUSSBAUM, Y., WULFF-NILSEN, C. Multiple-Source Multiple-Sink Maximum Flow in Directed Planar Graphs in Near-Linear Time. In: *Preceeding of the 52th Annual iEEE Symposium on Fundations of Computer Science,* New York: IEEE Press, 2011, p. 170-179. Disponibil: https://arxiv.org/pdf/1105.2228.pdf.
13. BOYCOV, Y., KOLMOGOROV, V. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. In: *IEEE Transactions on PAMI, 2004, vol. 26,* no. 9, p. 1124-1137. Disponibil: http://www.csd.uwo.ca/~yuri/Papers/pami04.pdf.
14. BUSACHER, R. G., GOWEN, P. J. *A procedure for determining a family of minimum-cost network flow patterns.* Bethesda, MD: Technical Report ORO-TP-15, In: Operations Research Office, The John Hopkins University, 1960. Disponibil: https://apps.dtic.mil/dtic/tr/fulltext/u2/249662.pdf.
15. CASAS, N. Genetic algorithms for multimodal optimization: A review. *arxiv:1508.05342v1 [cs.NET]*, 2015. Disponibil: https://arxiv.org/pdf/1508.05342.pdf.
16. CAUCHY, A. *Méthode générale pour la résolution des systèmes d'équations simultanées,.* Paris: BACHELIER, IMPRIMEUR -LIBRAIRE, 1847. Disponibil: https://gallica.bnf.fr/ark:/12148/ bpt6k2982c/f540.image.langEN.
17. CHANDRASIRI, A. M., & Samarathunge, D. M. Application of Minimum Cost Flow Problem: A Case Study of Crown Distributors in Kegalle, Sri Lanka. In: *International Journal of Scientific & Engineering Research , 2017, vol. 8,* no. 1, p. 1850-1853. Disponibil: https://www.researchgate.net/publication/313861666_Application_of_Minimum_Cost_Flow_Problem_A_Case_Study_of_Crown_Distributors_in_Kegalle_Sri_Lanka. ISSN 2229-5518.
18. CHEN, Q., SHI, F. Model for Microcirculation Transportation Network Design. In: *Mathematical Problems in Engineering, 2012, Article ID 379867, doi:10.1155/2012/379867 [citat august 2018]*, 2012, 11 pagini. Disponibil: http://downloads.hindawi.com/journals/mpe/2012/379867.pdf.
19. CHEN, D., Ni, S., XU, C., LV, H., WANG, S. High-Speed Train Stop-Schedule Optimization Based on Passenger Travel Convenience. In: *Hindawi, Mathematical Problems in Engineering, 2016, Article ID 8763589, http://dx.doi.org/10.1155/2016/8763589, [citat august 2018]*, 10 pagini. Disponibil: http://downloads.hindawi.com/journals/mpe/2016/8763589.pdf.

20. CHEN, Y., RILETT, L. R. Signal Timing Optimization for Corridors with Multiple Highway-Rail Grade Crossings Using Genetic Algorithm. In: *Journal of Advanced Transportation, 2018, Article ID 9610430, https://doi.org/10.1155/2018/9610430*, *[citat august 2018]*, 14 pagini. Disponibil: http://downloads.hindawi.com/journals/jat/2018/9610430.pdf.
21. CHERIYAN, J., LAEKHAUKIT, B. Aproximation algorithms for minimum-cost k-(S,T) connected digraphs. In: *SIAM J Discete Math., 2013, vol. 27,* no. 3, p. 1450-1481. Disponibil: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.640.3797&rep=rep1&type=pdf.
22. CIUPALĂ, L. The minimum cost flow problem with surplus. In: *Bulletin of the Transilvania University of Braşov, 2010, vol. 3,* no. 52, p. 177-182. ISSN 2065-2151.
23. CIUPALĂ, L. A generic preflow algorithm for maximum flow in semibipartite networks. In: *Bulletin of the Transilvania University of Braşov, 2014, vol. 7,* no. 56, p. 103-108. ISSN 2065-2151.
24. CIUPALĂ, L. Wave algorithm for maximum flow in semi-bipartite networks. *Bulletin of the Transilvania University of Braşov, 2016, vol. 9,* no. 58, p. 111-118. ISSN 2065-2151.
25. COHEN, M. B., MADRY, A., SANKOWSKI, P., VLADU, A. Negative-weight shortest paths and unit capacity minimum cost flow in O(m[superscript 10/7]log W) time. *28th Annual ACM-SIAM Symposium on Discrete Algorithm, 16-19 January 2017,* p. 752-771, Barcelona, Spain: Association for Computing Machinery. Disponibil: http://hdl.handle.net/1721.1/113883.
26. CORLAT, S., CORLAT, A. *Grafuri. Noţiuni. Algoritmi. Implementări.* Chişinău, 2012.
27. CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. *Introduction to Algorithms* (ed. 2nd Edition). London, England: MIT Press, 2002. ISBN 0-262-03293-7.
28. COUETOUX, B., DAVIS, J. M., WILLIAMSON, D. P. A 3/2-aproximation algorithm for some minimum-cost graph problems. *Math. Progr. B, 2015, vol. 150,* no. 1, p. 19-34. https://doi.org/10.1007/s10107-013-0727-z.
29. CRISTIANO, P., KELNER, J. A., MADRY, A., SPIELMAN, D. A., TENG, S.-H. Electrical Flows, Lapcacian Systems, and Faster Aproximation of Maximum Flow in Undirected Graphs. In: *Preceeding of the Annual ACM Symposium on Theory of Computing, 2011,* p. 273-282, New York: ACM Press. https://dl.acm.org/doi/pdf/10.1145/1993636.1993674.
30. CUNNINGHAM, W. H. A Network Simplex Method. In: *Mathematical Programmong*, 1976, no. *11*, p. 105-116. Disponibil: https://www.researchgate.net/publication/225960770_A_Network_Simplex_Method.

31. DANTZIG, G. B. Aplication of the Simplex Method to a Transportation Problem. *Activity of Analysis Production and Allocation,* T. C. Koopmans, Ed., John Wiley and Sons, New York, *1951,* p. 359-373.
32. DANTZIG, G. B. *Linear Programming and Extensions.* Princeton, NJ: Princeton University Press, 1963. Disponibil: https://www.rand.org/content/dam/rand/pubs/reports/2007/R366part1.pdf.
33. DAVIS, J. M., WILLIAMSON, D. P. A dual-fitting 3/2-aproximation algorithm for some minimum-cost graph problems. In: *20th Annual European Symposium, no. 7501 in Lecture Notes in Computer Science, 2012,* p. 373-382, Springer. https://doi.org/10.1007/978-3-642-33090-2_33.
34. DAWUNI, M., DARKWAH, K. F. Maximum flow-minimum cost algorithm of a distribution company in Ghana: Case of 'NAAZO' Bottling Company, Tamale, Tamale Metropolis. In: *African Journal of Mathematics and Computer Science Research, 2015, no.8* (2), p. 23-30. ISSN 2006-9731. DOI: 10.5897/AJMCSR2014.0569. Disponibil: https://pdfs.semanticscholar.org/d49e/69c56bb7f853d166218ab81d52c8d7fa4fbf.pdf.
35. DIAZ-PARA, O., RUIZ-VANOYE, A., LORANCA, B. B., FLUENTES-PENNA, A., BARRERA-CAMARA, R. A. A survey of transportation problem, In: *HPC J. of AM*, 2014, ID 848129, 17 pages, http://dx.doi.org/10.1155/2014/848129 [citat Iulie 2018]. Disponibil: http://downloads.hindawi.com/journals/jam/2014/848129.pdf.
36. DING, S. Uncertain minimum cost flow problem. In: *Soft. Comput., 2014, no. 18*, p. 2201-22017. DOI: 10.1007/s00500-013-1194-4. Disponibil: https://www.researchgate.net/publication/259633789_Uncertain_minimum_cost_flow_problem.
37. DINIC, E. A. Algorithm for solution of a problem of maximum flow in networks with power estimation. In: *Soviet Mathematics Doklady, 1970,* no. 11, p. 1277-1280.
38. DJAMEL, A., AMEL, N., HOAI, L. T., AHMED, Z. A modified clasical algorithm ALPT4C for solving a capacitated four-index transportation problem. In: *ACTA Mathematica Vietnamica, 2012, vol. 37,* no. 3, p. 379-390. Disponibil: http://journals.math.ac.vn/acta/images/stories/pdf1/Vol_37_No_3/Bai6_Dja_Amel_An_Ahmed_Acta_11_51.pdf.
39. EBRAHIMNEJAD, A., NASSERI, S. H. Extension of network primal simplex algorithm for solving minimum cost flow problem with fuzzy costs based on ranking functions. In: *Annals of Fuzzy Mathematics and Informatics, 2012, vol. 4,* no. 1, p. 9-24. ISSN 2093–9310. Disponibil: http://www.afmi.or.kr.

40. EDMONDS, J., KARP, R. M. Theoretical improvements in algoritmic efficiency for network flow problems. În: *Combinatorial Structures and Their Applications, 1970,* p. 93-96. New-York: Gordon and Breach.
41. EDMONDS, J., KARP, R. Theoretical improvements in algorithmic efficiency for network flow problems. In: *Journal of the ACM, 1972, no.* 19, p. 248-264. Disponibil: https://web.eecs.umich.edu/~pettie/matching/Edmonds-Karp-network-flow.pdf.
42. EL-SHERBENY, N. A. Minimum Cost Flow Time-Windows Problem with Interval Bounds and Flows. In: *Theoretical Mathematics & Applications, 2016, no. 6* (3). Print: ISSN 1792-9687, Online 1792-9709. Disponibil: http://www.scienpress.com/Upload/TMA/Vol%206_3_2.pdf.
43. ERICKSON, J., FOX, K., & LKHAMSUREN, L. Holiest Minimum-Cost Paths and Flows in Surface Graphs. *Cornell University Library,* 2018. Disponibil: https://arxiv.org/pdf/1804.01045.pdf.
44. EROGLU, E., ADIGUZEL, B. A genetic algorithm based approach to the workload balancing problem. In: *4th International Logistics and Supply Chain Management Congress,* 2006, p. 251-257. Disponibil: https://www.researchgate.net/publication/329916345_A_Genetic_Algorithm_Based_Approach_to_the_Workload_Balancing_Problem_4th_International_Logistics_and_Supply_Chain_Management.
45. FLYNN, M. J. Some computer organization and their affectiveness. 1972, no. *9*, p. 948 - 960. DOI: 10.1109/TC.1972.5009071.
46. FONTES, D., GONCALVES, J. F. Heuristic solutions for general concave minimum cost network flow problems. In: *Networks, An International Journal, https://doi.org/10.1002/net.20167, [citat August 2018], 2007, vol. 50,* no. 1, p. 67-76.
47. FORD, L. R., FULKERSON, D. R. Maximal flow through a network. In: *Canadian Journal of Mathematics, 1956,* no. 8, p. 399-404. Disponibil: http://www.cs.yale.edu/homes/lans/readings/routing/ford-max_flow-1956.pdf.
48. FORD, L. R., FULKERSON, D. R. *Flows in networks.* New Jersy, USA: The RAND Corporation, Pricenston University Press, 1962.
49. FULKERSON, D. R. An Out-of-Kilter Method for Minimal-Cost Flow Problems. In: *Journal of the Society for Industrial and Applied Mathematics, 1961, vol. 9,* no. 1, p. 18-27. Disponibil: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.228.7120&rep=rep1&type=pdf.
50. GALIL, Z., NAAMAD, A. An O(EVlog2V) Algorithm for the Maximal Flow Problem. In: *Journal of Computer and System Sciences, 1980, no. 21*, p. 203-217. Disponibil: https://core.ac.uk/download/pdf/81946904.pdf.

51. GAMEȚCHI, A., SOLOMON, D. *Cercetări operaționale*, Vol. II, 2015, Chișinău: Evrica.
52. GERANIS, G., PAPARIZZOS, K., SIFALERAS, A. A dual exterior point simplex type algorithm for the minimum cost network flow problem. In: *Yugoslav Journal of Operations Research, 2009, vol. 19,* no. 1, p. 157-170. DOI:10.2298/YUJOR0901157G. Disponibil: http://yujor.fon.bg.ac.rs/index.php/yujor/article/view/319/210.
53. GHAHERI, A., SHOOR, S., NADERAN, M., HASEINI, S. S. The aplicants of Genetic Algorithms in Medicine. In: *Oman Med. J., 2015, vol. 30,* no. 6, p. 406-416. Disponibil: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4678452/pdf/OMJ-D-15-00162.pdf.
54. GHIYASVAND, M. An O(m(m+nlogn)log(nC)) - time algorithm to solve the minimum cost tension problem. In: *Theoretical Computer Science*, 2012a, no. 448, p. 47-55. DOI: 10.1016/j.tcs.2012.03.042.
55. GHIYASVAND, M. Solving the minimum flow problem with interval bounds and flows. In: *Sadhana, Indian Academi of Sciences, 2012b, vol. 37,* no. 6, p. 665-674. https://doi.org/ 10.1007/s12046-012-0100-6.
56. GOLDBERG, A. V., TARJAN, R. E. Solving minimum-cost flow problems by successive approximations. In: *19th ACM Symposium on Theory of Computing, STOC 87, 1987,* p. 7-18. New York: ACM Press.
57. GOLDBERG, A. V., TARJAN, R. E. A New Approach to the Maximum-Flow Problem. In: *Journal of the Association for Computing Machinery, 1988, vol. 35,* no. 4, p. 921-940. Disponibil: https://dl.acm.org/doi/pdf/10.1145/48014.61051.
58. GOLDBERG, A. V., TARJAN, R. E. Finding Minimum-Cost Circulations by Canceling Negative Cycles. In: *Journal of the Association for Computing Machinery, 1989, vol. 36,* no. 4, p. 873-886. Disponibil: https://dl.acm.org/doi/pdf/10.1145/76359.76368.
59. GOLDBERG, A. V., TARJAN, R. E. Finding minimum-cost circulations by successive aproximation. In: *Math. Oper. Res.*, 1990, vol. 15, no. 3, p. 430-466.
60. GOLDBERG, A. V., TARJAN, R. E. Eficient maximum flow algorithms. In: *Communications of the ACM, 2014, vol. 57,* no. 8, p. 82-89.
61. GOLDBERG, A. V., KAPLAN, H., HED, S., TARJAN, R. E. Minimum Cost Flow in Graphs with Unit Capacities. In: *32th Symposium on Theoretical Aspects of Computer Science,* 2015, p. 406-419.
62. GOLDBERG, A. V., HED, S., KAPLAN, H., TARJAN, R. E. Minimum-Cost Flows in Unit-Capacity Network. In: *Theory of Computing System, 2017, vol. 61,* no. 4, p. 987-1010.
63. GRIGORIADIS, M. D. An efficient implementation of the network simplex method. In: *Mathematical Programming Studies, 1986, no. 26*, p. 83-111.

64. GUO, H., WANG, X., ZHOU, S. (2015). A Transportation Problem with Uncertain Costs an Random Supplies. In: *International Journal e-Navigation and Maritime Economy, 2015, no. 2*, p. 1-11. http://dx.doi.org/10.1016/j.enavi.2015.06.001.
65. HAKIM, M. A., KABIR, M. R. An Efficient Approach for Finding an Initial Basic Feasible Solution for Transportation Problems, In: *Progress in Nonliniar Dynamics and Chaos, 2017, vol. 5,* no. 1, p. 17-23. DOI: http://dx.doi.org/10.22457/pindac.v5n1a3, ISSN: 2321 – 9238 (online).
66. HALEY, K. B. The solid transportation problem. In: *Operat. Research, 1962, no. 10*, p. 448-463.
67. HAN, S., PENG, Z., WANG, S. The maximum flow of uncertain network. In: *Information Sciences, 2014, no. 265, p. 167-175. http://dx.doi.org/ 10.1016/j.ins.2013.11.029* .
68. HE, Q., SHABBIR, A., NEMHAUSER, G. L. Minimum Concave Cost Flow Over a Grid Network. In: *Mathematical Programming, 2015, vol. 150,* no. 1, p. 79-98.
69. HITCHCOCK, F. L. The distribution of a Product from Several ouces to Numerous Localities. In: *Lournal of Math. Phys, 1941, no. 20* (1-4), p.224-230.
70. HOLLAND, J. H. *Adaptation Systems for Engineers and Scients.* University Michigan, 1975.
71. HOLLAND, J. H. Genetic algorithms. In: *Scientific american, 1992, no. 267*, p. 66-72.
72. HOLZHAUSER, M., KRUMKE, S. O., THIELEN, C. Maximum flows in generelized processing networks. In: *Journal Comb. Optim, 2017, vol. 33,* no. 4, p. 1226-1256.
73. HOOPGOOD, A. A. *Inteligent Systems for Engineers and Scientists* (ed. 3rd). CRC Press, Taylor and Francis Group. 2012, 451p. ISBN 9781439821206.
74. HORST, R., PARDALOS, P. M. *Handbook of Global Optimization.* Springer - Science + Business Media, 1st edition, 1995, 880p. ISBN-10: 1461358388.
75. HORST, R., TUY, H. *Global optimization, Deterministic Approaches.* New-York: Springer-Verlag, 3rd edition, 1996, 735p. ISBN 978-3-662-03199-5. DOI 10.1007/978-3-662-03199-5
76. IRI, M. A new method for solving transportation-network problems. In: *Journal The Operations Research Society of Japan, 1960, no. 3*, p.27-87. Disponibil: http://www.orsj.or.jp/~archive/pdf/e_mag/Vol.03_01_02_027.pdf.
77. JIANG, Y., XU, X., ZHANG, H., LUO, Y. Taxiing Route Scheduling between Taxiway and Runway in Hub Airport. In: *Mathematical Problem in Engineering, 2015, Article ID 925139, http://dx.doi.org/10.1155/2015/925139 [citat August 2018]*, 14 pagini. Disponibil: http://downloads.hindawi.com/journals/mpe/2015/925139.pdf.

78. JUNGINGER, W. On reprezentative of multi-index transportation problems. In: *European Journal of Operational Research, 1993, no. 66*, p. 353-371. DOI:10.1016/0377-2217(93)90223-A.
79. KANTOROVICI, L. On the translocation of masses. In: *Acad. Scientific URSS*, 1958, vol. 5 no. 1, p. 199-201.
80. KARZANOV, A. V. Determining the maximal flow in a network by the method of preflows. In: *Dokl. Akad. Nauk SSSR, 1974, vol. 215, no. 1*, p. 49-52. Disponibil: http://www.mathnet.ru/links/2f37bf24539ed692d5d88dc0f24bf6ca/dan38151.pdf.
81. KELNER, J. A., LEE, Y. T., ORECCHIA, L., SIDFORD, A. An Almost-Linear-Time Algoritm for Aproximate Max Flow in Undirected Graphs, and its Multicomodity. In: *Proceedings of the Twenty-Fifth Annual ACM - SIAM Symposium on Discrete Algorithms, 2014,* p. 217-226. Philadelphia: SIAM. https://doi.org/10.1137/1.9781611973402.16, ISBN: 978-1-61197-338-9.
82. KENNETH, D. J. *An Analysis of the Behaivor of a Class of Genetic Adaptive Systems.* Doctoral thesis, Depart. of Computer and Communication Sciences, Univ. of Michigan, 1975. 271p.
83. KHURANA, A., ALDAKHA, V., LEV, B. *Multi-index constrained transportation problem with bounds on availabilities.* In: Operations Research Perspectives, 2018, vol. 5, p. 319-333, [citat ianuarie 2020], Disponibil: https://reader.elsevier.com/reader/sd/pii/S2214716018300666?token=D790D4B84992E239FD60B7003CA95CD47B1F6168FA4A7C8AC2752A9D896EBDE8A12DA35C97E7BDAAF57231322EA7DF87, doi.org/10.1016/j.orp.2018.10.001.
84. KIRALY, Z., KOVAS, P. Efficient implementations of minimum-cost flow algorithms. In: *Acta Univ. Sapientiae, Informatica, 2012, vol. 4, no.* 1, p. 67-118.
85. KLANSEC, U., PSUNDER, M. *Solving the nonlinear transportation problem by global optimization*. [citat ianuarie 2020], 2010, vol. 25, no. 3, p. 314-324. ISSN 1648-4142 print / ISSN 1648-3480 online, doi: 10.3846 / transport.2010.39.
86. KLEIN, M. A primal method for minimal cost flows with applications to the assignment and transportation problems. In: *Managment science, 1967, vol. 14,* no. 3, p. 205-220. https://doi.org/10.1287/mnsc.14.3.205.
87. KOOPMANS, T. C. Measurement Without Theory. In: *The review of Economics and Statistics, 1947, vol. 29,* no. 3, p.161-172. Disponbil: http://static.stevereads.com/papers_to_read/koopmans_review_of_measuring_business_cycles.pdf.

88. KOVACS, P. Minimum-cost flow algorithms: An experimental evaluation. In: *Egevary Research Group, Teechnical reports, 2013, no. 4, p. 94-127*. Disponibil: https://web.cs.elte.hu/egres/tr/egres-13-04.pdf. ISSN 1587–4451.
89. KUDJO, P. K., OCQUAYE, E. Review of Genetic Algorithm and Application in Software Testing. In: *International Journal of Computer Applications, 2017, vol. 160,* no. 2, p. 1-6. ISSN 0975-8887.
90. KUHN, H. W., TUCKER, A. W. Nonlinear Programming. In: *Second Berkeley Symposium on Mathematical Statistics and Probability, 1951,* p. 481 - 492. Berkeley: University of California Press. Disponibil: http://web.math.ku.dk/~moller/undervisning/MASO2010/kuhntucker1950.pdf.
91. LI, H.-L., YU, C.-S. A global optimization method for nonconvex separable programming problems. In: *European Journal of Operational Research, 1999, vol. 117*, no. 2, p. 275-292. Disponibil: https://ir.nctu.edu.tw/bitstream/11536/31124/1/000081318000007.pdf.
92. LIU, T.-K., LIN, S., HSUEH, P.-W. Optimal design for transport and logistics of steel mill by product based on double-layer genetic algorithms. In: *Journal of Low Frequency Noise, Vibration and Active Control*, *2019*, p. 1 - 22. *doi.org/10.1177/146134841872368*. Disponibil: https://journals.sagepub.com/doi/pdf/10.1177/1461348419872368.
93. LOBEL, A. Solving Large-Scale Real-Wold Minimum-Cost Flow Problems by a Network Smplex Method. *Technical Report SC 96-7, Zuse Institute Berlin*, *1996*. Disponibil: https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/218.
94. LOZOVANU, D., **PASHA, T.** An algorithm for solving the transport problem on network with concave cost functions of flow on edges. In: *Computer Science Journal of Moldova, 2002, vol. 10,* no. 3(30), p. 341-347. ISSN 1561-4042.
95. LUENBERGER, D. G., YE, Y. Linear and nonlinear programming. In: *International Series in Operations Research and management science, 2008,* 555 p. Stanford: Springer. ISBN: 978-0-387-74502-2.
96. MAHER, R. A., ABDULA, F. A. An Algorithm for Cost-Minimizing in Transportation via Road Networks Problem. In: *International Journal of Mathematical and Computational Methods, 2017, vol. 2*, p. 292-299. Disponibil: https://pdfs.semanticscholar.org/e5db/22d56886c87b0a986d295fa3d7e9b3ae2901.pdf.
97. MARDY, A. Navigating central path with electrical flows From flows to matchings and back. In: *Prec. of the Annual IEEE Symposium on Foundations of Computer Science, 2013,* p. 253-262. New York: IEEE Press. Disponibil: https://arxiv.org/abs/1307.2205v3.

98. MADRY, A. Computing Maximum flow with Augmenting Electrical Flows. In: *57th Annual IEEE Sympposium om Fondations of Computer Science - FOCS 2016, 2016,* p. 593-602. Los Alamiotos, CA: IEEE Comp. Soc. https://people.csail.mit.edu/madry/ docs/aug_flow.pdf.

99. MEGIDDO, N. Optimal flow in networks with multiple sources and sinks. In: *Mathematical Programming, 1974, vol. 7*, no. 1, p. 97-1007. Print ISSN 0025-5610, Online ISSN 1436-4646

100. MEHTA, R. A New Push-Relabel Algorithm for Sparse Networks. In: *Computer Science, Data Structures and Algorithms*, *2014.* Disponibil: arxiv.org/pdf/1310.7840.pdf.

101. MILLER, C. E. The simplex Method for Local Separable Programming. In: *Recent Advances in Mathematical Programming*, *1963*.

102. MILLER, G. L., NAOR, J. Flow in planar graphs with multiple sources and sinks. In: *SIAM Journal on Computing, 1995, vol. 24,* no. 5, p. 1002-1017. DOI: 10.1109/SFCS.1989.63464. Print ISSN 0097-5397, Online ISSN 1095-7111.

103. MINOUX, M. Solving integer minimum cost flows with separable convex cost objective polynomially. *Mathematical Programming Studies, 1986, vol. 26*, p. 237-239. DOI: 10.1007/BFb0121104.

104. MIZUNO, S., SUKEGAWA, N., DEZA, A. A primal-simplex based Tardos' algorithm. In: *Operations Research Letters, 2014, vol. 43,* no. 6. p. 625-628. Disponibil: http://www.optimization-online.org/DB_FILE/2014/09/4529.pdf.

105. MOANȚĂ, D. Principii privind algoritmii genetici pentru soluționarea unei probleme tridimensionale de transport. In: *Revista Informatica economică, 1998,* no. 6, p. 47-51.

106. MONGE, G. *Mémoire sur la théorie des déblais et de remblais.* Histoire de l'Académie Royale des Sciences de Paris, avec les Mémoires de Mathém. et de Physique pour la même année, 1781. Disponibil: https://images.math.cnrs.fr/Gaspard-Monge,1094.html?lang=fr.

107. NINH, P.-X. N index transportation problem. In: *Mathematical Journal, 1979, vol. 7,* no. 1, p. 18-25.

108. ORLIN, J. B., PLOTKIN, S. A., TARDOS, E. Polinomial dual network simplex algorithms. *Mathematical Programming, 1993, no. 60*, p. 255-276. https://doi.org/10.1007/BF01580615.

109. ORLIN, J. B. A faster strongly polinomial minimum cost flow algorithm. In: *20th ACM Symposium on Theory of Computing, STOC 88, 1988,* p. 377-387. New York: ACM Press. ISBN: 978-0-89791-264-8.

110. ORLIN, J. B. Max Flows in O(nm) Time, or Better. In: *Proceedings of the 45th Annual ACM Symposium on Theory of Computing - STOC '13,* June 1-4, 2013, Palo Alto, California, USA. p.765-774. ISBN 9781450320290.

111. OSABA, E., CARBALLEDO, R., DIAZ, F., ONIEVA, E., IGLESIA, I., PERALLOS, A. Crossover versus Mutation: A Comparative Analyeie of the Evolutionary Strategy of Genetic Algorithms Applied to Combinatorial Optimization Problem. In: *The Scientific World Journal, 2014. Article ID 154676, http://dx.doi.org/10.1155/2014/154676, [citat august 2018]*, 22 pages. Disponibil: http://downloads.hindawi.com/journals/tswj/2014/154676.pdf.

112. **PAȘA, T.** Proprietățile soluționării optimale a problemei neliniare de transport pe rețea. În: *Analele Universităţii de Stat din Moldova, Seria "Ştiinţe fizico-matematice", 2004,* p. 134-140. ISSN 1811-2641.

113. **PAȘA, T.** Algoritmul determinării soluțiilor admisibile și a soluțiilor optime pentru problemele de transport pe rețea cu o singură sursă. In: *Analele Universităţii de Stat din Moldova, Seria "Ştiinţe fizico-matematice", 2005,* p. 188-190. ISSN 18811-2641.

114. **PAȘA, T.** Sinteza metodelor de soluționare a unor cazuri particulare ale problemei de transport pe rețea. În: *Studia Universitatis Moldaviae, Seria Științe Exacte, 2011, nr.7* (47), p. 53-59. Online ISSN 2345-1033.

115. **PAȘA, T.**, UNGUREANU, V. Wolfram Mathematica as an enviroment for solving concave network transportation problem. In: *The Fourth Conference of Mathematical Society of the Republic of Moldova dedicated to the centenary of Vladimir Andrunachevici (1917 - 1997), 28 iunie - 2 iulie, 2017a*, p. 429 - 432. Chișinău: IMCS, AȘM. ISBN 978 - 9975 - 71 - 915 - 5.

116. **PAȘA, T.**, UNGUREANU, V. Asupra metodei potențialelor pentru problema de transport degenerată. *Studia Universitas Moldaviae, Seria Științe Exacte, 2017b, nr. 2* (102), p. 30-36. ISSN 1857 – 2073.

117. **PAȘA, T.** Problema fluxului maxim în rețele – analiza și sinteza algoritmilor de soluționare. *Studia Universitatis Moldaviae, Seria „Științe exacte", 2017c, nr. 7* (107), p. 150-158. ISSN 1857-2073, Online ISSN 2345-1033.

118. **PAȘA, T.**, UNGUREANU, V. Non-Linear Concave Transportation Problem Solving and Implementation using Wolfram Language., In: *ITSN, 2017d,* p. 30-39. Chișinău. ISBN 978-9975-3168-5-9.

119. **PAȘA, T.**, UNGUREANU, V. Solving the transportation problem with piecewise - linear concave cost function. In: *Review of AFA, The Scientific Informative Review, 2017e, vol. XV no. 2 (34), p. 49 -56. SPSR, DOI: 10.19062/1842-9238.2017.15.2.* ISSN 1842-9238.

120. **PAȘA, T.**, UNGUREANU, V. Applying sequential and parallel programming to solve a non-linear transport problem. In: *ICMCS, October 19 - 21, 2017f*, p. 247-251. Cișinău, Republic of Moldova. ISBN 978-9975-4264-8-0.

121. **PAȘA, T.** Soluționarea problemei de transport pe rețea ca problemă a programării nelineare. În: *Studia Universitatis Moldaviae, Seria Științe Exacte, 2018a, nr. 7* (117), p. 14-24. ISSN 1857 - 2073.

122. **PAȘA, T.** Multi-index transport problem with non-linear cost functions. In: *Romai J., 2018b, vol. 14,* no. 2, p. 129-137. Disponibil: https://rj.romai.ro/arhiva/2018/2/Pasa.pdf.

123. **PAȘA, T.** The genetic algorithm for solving the non-linear transportation problem. In: *Review of the Air Force Academy, The Scientific Informative Review, 2018c,* nr. 2 (37), p. 37 - 44. Online ISSN: 2069-4733, ISSN-L: 1842-9238.

124. **PAȘA, T.**, The application of parallel programming in solving the non-linear transport problem, *Conferința Internațională Modelare Matematică, Optimizare și Tehnologii informaționale, 12-16 noiembrie, 2018d, Chișinău, p. 164-168*. ISBN 978-9975-62-421-3.

125. **PAȘA, T.**, UNGUREANU, V. Solving the non-linear 4-index transportation problem. In: *The Fifth Conference of Mathematical Society of the Republic of Moldova, 2019a,* p. 221-224. Chișinău: Vladimir Andrunachievici Institute of Mathematics and Computer Science. Disponibil: https://ibn.idsi.md/sites/default/files/imag_file/221-224_9.pdf.

126. **PAȘA, T.** Solving non-linear multi-index transportation problems. In: *Romai J., 2019b, vol. 15,* no. 2, p. 91-99. Disponibil: https://rj.romai.ro/arhiva/2019/2/Pasa.pdf.

127. **PAȘA, T.** Solving transportation problems with concave cost functions using genetic algorithms. In: *Computer Science Journal of Moldova, 2020a, vol. 28 no. 2 (83),* p.140-151.

128. **PAȘA, T.**, Algoritmi de soluționare a problemelor neliniare de transport cu mai mulți indici. În: *Studia Universitatis Moldaviae, Seria Științe Exacte, 2020b, nr. 2* (132), p. 36-44. ISSN 1857 - 2073 Online ISSN 2345-1033.

129. **PAȘA, T.** Genetic algorithm for solving transportation problems on networks with one source and multiple sinks. In: *ITM Web Conf., ICAMNM 2020, Section: Applied Mathematics and Numerical Methods, 2020c, nr. 34*, 11 pages. https://doi.org/10.1051/itmconf/20203402006, Online ISSN 2271-2097.

130. PAIU, O. I., DUMITRESCU V. Algoritmi genetici seriali și paraleli. In: *Informatică economică, 1998,* vol. 6, p. 52-58. Disponibil: http://revistaie.ase.ro/content/6/52-58.pdf.

131. PHAM, T.-H., DOTT, P. Four indexes transportation problem with interval cost parameter for goods allocation planning. In: *LINDI 2012, 4th IEEE International Symposium on Logistics and Industrial Informatics, September 5-7, 2012,* pp. 87-92. Smolenice, Slovakia. Disponibil: https://www.ixueshu.com/document/ee30c0f624fdb0ac9e908a3c0d3d464d.html.

132. PHAM, T.-H., DOTT, P. An Exact Method for Solving Four Index transportation Problem and Industrial Application. In: *American J. of Operational Research, 2013, vol. 3,* no. 2, p. 28-44. DOI: 10.5923/j.ajor.20130302.02. Print ISSN: 2324-6537, Online ISSN: 2324-6545.

133. ROCK, H. Scaling techniques for minimal cost network flows. In: *Discrete Structures and Algorithms*, 1980.

134. ROSEN, J. B. The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints. In: *J. of the Society for Industrial and Applied Mathematics, 1960, vol. 8,* no. 1, p. 181-217. Print ISSN 0368-4245, Online ISSN 2168-3484.

135. ROSTAMI, R., EBRAHIMNEJAD, A. An approximation algorithm for discrete minimum cost flows over time problem. In: *Int. J. Operational Research, 2014, vol. 20,* no. 2, p. 226-239. Online ISSN 1745-7653, Print ISSN 1745-7645.

136. SADEGHEIH, A., DRAKE, P. R. A novel experimrntal analysis of the minimum cost flow problem. In: *IJE Transactiona A: Basics, 2009, vol. 22,* no. 3, p. 251-268. Disponibil: http://www.ije.ir/article_71798_bcca1a25e39e1d8a3e7fe233b4807baf.pdf.

137. SARODE, M. V. Application of a Simplex Method to Find the Optimal Solution. In: *International Journal of Innovations of Engineering and Science, 2017, vol. 2,* no. 2, p. 21-24. ISSN 2456 - 3463.

138. SENAPATI, S. Multi-index bi-criterion transportation problem. A fuzy aproach. In. *International Journal of Advanced Engineering, Management and Science, 2018, vol. 4,* no. 7, p. 550-556. https://dx.doi.org/10.22161/ijaems.4.7.8. Online ISSN 2454-1311.

139. SHENG, S., DECHEN, Z., XIAOFEI, X. Genetic Algorithm for the Transportation Problem with Discontinuous Piecewise Linear Cost Function. In: *IJCSNS, 2006, vol. 6,* no. 7A, p. 182-190. ISSN : 1738-7906.

140. SHERMAN, J. Nearly Maximum Flows in Nearly Linear Time. In: *Proceeding at the annual IEEE Symposium on Fondations of Computer Science - FOCS 2013,* p. 263-269. Los Alamitos, CA: IEEE Computer Soc. https://doi.org/10.1109/FOCS.2013.36.

141. SIFALERAS, A. Minimum cost network flows: problems, algorithms, and software. In: *Yugoslav Journal of Operations Research, 2013, vol. 23,* no. 1, p. 3-17. DOI: 10.2298/YJOR121120001S.

142. SINGH, S., CHAUHEN, S. K., KULDEEP, S. K. A bi-criteria multi-index bulk transportation problem. In. *Annals of Pure and Applied Mathematics*, *2018, vol. 16, no. 2, 2018, p. 479-485* [citat ianuarie 2020], http://dx.doi.org/10.22457/apam.v16n2a26. http://www.researchmathsci.org/APAMart/apam-v16n2-26.pdf. Print ISSN: 2279-087X, Online ISSN 2279-0888.

143. SLEATOR, D. D., TARJAN, R. E. A data structure for dinamic trees. In. *Journal of Computer and System Sciences, 1983, vol. 26,* no. 3, p. 362-391. Disponibil: https://www.cs.cmu.edu/~sleator/papers/dynamic-trees.pdf.

144. SLEATOR, D. D., TARJAN, R. E. Self-Adjusting Binary Search Trees. In. *Journal of the Association for Computing Machinery, 1985, vol. 32,* no. 3, p. 652-686. Disponibil: https://www.cs.cmu.edu/~sleator/papers/self-adjusting.pdf.

145. SOKKALINGAM, P. T., AHUJA, R. K., ORLIN, J. B. New polynomial-time cycle-canceling for minimum cost flows. In. *Networks, 2000, vol. 36*, no. 1, p. 53-63. https://doi.org/10.1002/1097-0037(200008)36:1<53::AID-NET6>3.0.CO;2-Y.

146. STANCU-MINASIAN, I. M, *Fractional programming: theory, methods, and applications*, Kluwer Academic Publishers, Dordrecht; Boston, 1997.

147. SUN, W., YUAN, Y.-X. Optimization theory and methods*,* In: *Nonlinear Programming, 2006, vol. 1.* Springer Science + Business Media, LLC. ISBN-10: 0-387-24975-3.

148. SURAKHI, O. M., QATAWNEH, M., OFEISHAT, H. A. A Parallel Genetic Algorithm for Maximum Flow Problem. *International Journal of Advanced Computer Science and Aplications, 2017, vol. 8,* no. 6, p. 159-164. Online ISSN 2156-5570, Print ISSN 2158-107X.

149. ȘCHIOPU, C. The maximum flows in bipartite dynamic networks. The static approach. In: *Annals of the University of Craiova, Mathematics and Computer Science Series*, 2016, *vol. 43, no. 2*, p. 200-209. ISSN 1223-6934. Disponibil: https://pdfs.semanticscholar.org/59a6/6d3ff47a176ec195caf62ff527e03dd0bd78.pdf.

150. TARJAN, R. E. Efficiency of the Primal Network Simplex Algorithm for the Minimum Cost Circulation Problem. In: *Math. Oper. Res., 1991, vol. 16*, no. 2. p. 272-291.

151. TARJAN, R., WARD, J., ZHANG, B., ZHOU, Y., MAO, J. Balancing Applied to Maximum Network Flow. In*: Prodeeding of the 14th European Symposium on Algorithms, 2006,* pg. 612-623. Berlin: Spriger-Verlag. ISBN 978-3-540-38875-3.

152. THOMAS, D., KOVOOR, B. C. A genetic algorithm approach to automous smart vehicle. Parking system. *Procedia Computer Science, 2018, vol. 125,* p. 68-76. https://doi.org/10.1016/j.procs.2017.12.011.

153. TOMIZAWA, N. On some techniques useful for solution of transportation network problems. *Networks, 1971, vol. 1*, no. 2, 173-194. https://doi.org/10.1002/net.3230010206.

154. TRANDAFIR, R. *Modele și algoritmi de optimizare.* București: AGIR, 2004, 252. ISBN 973-8466-76-8.

155. VEGH, L. A. Strongly polynomial algorithm for a class of minimum - cost flow problems with separable convex objectives. In: *Proceedings of 44th symposium on Theory of Computing, 2012,* p. 27-40. New York, USA: ACM. ISBN: 978-1-4503-1245-5.

156. WANG, I.-L., LIN, S.-J. A network simplex algorithm for solving the minimum distribution cost problem. In: *Journal of infustrial and management optimization, 2009, vol. 5,* no. 4, p. 929-950. Print ISSN 1547-5816, Online ISSN 1553-166X.

157. WEINTRAUB, A. A Primal Algorithm to Solve Network Flow Problems with Convex Costs. In: *Management Science, 1974, vol. 21,* no. 1, p. 87-97. https://doi.org/10.1287/mnsc.21.1.87.

158. WOLFRAM, S. *An elementary introduction to the Wolfram Language* (ed. 1-st edition). Friesens, Canada: Manitoba. 2016, 324 p. Print ISBN 978-1-944183-00-4, Online ISBN 978-1-944183-01-1.

159. YANG, Z., WANG, W., CHEN, S., DING, H., LI, X. Genetic Algorithm for Multiple Bus Line Coordination on Urban Arterial. In: *Computational Inteligence and Neuroscience 2015, Article ID 868521, http://dx.doi.org/10.1155/2015/868521 [citat august 2018]*, 7 pagini. Disponibil: http://downloads.hindawi.com/journals/cin/2015/868521.pdf.

160. ZHANG, P., SUN, Z., LIU, X. Optimized Skip-Stop Metro Line Operation Using Smart Card Data. In: *Hindawi, Journal of Advanced Transportation, 2017. Article ID 3097681, https://doi.org/10.1155/2017/3097681, [citat august 2018]*, 17 pages. Disponibil: http://downloads.hindawi.com/journals/jat/2017/3097681.pdf.

161. ZHANG, E., MEI, Q., LIU, M., & ZHENG, F. Stowage Planning in Multiple Ports with Shifting Fee Minimization. In: *Hindawi, Scientific Programming, 2018. Article ID 3450726, https://doi.org/10.1155/2018/3450726 [citat august 2018]*, 9 pagini. Disponibil: http://downloads.hindawi.com/journals/sp/2018/3450726.pdf.

162. ZITOUNI, R., KERAGHEL, A., BENTERKI, D. Elaboration and Implementation of an Algorithm Solving a Capacitated Four-Index Transportation Problem. *Applied Mathematical Sciences, 2007, vol. 1,* no. 53, p. 2643-2657. Disponibil: https://www.researchgate.net/publication/267118025_Elaboration_and_implantation_of_an_algorithm_solving_a_capacitated_four-index_transportation_problem.

163. ZITOUNI, R., ACHACHE, M. A numerical comparison between two exact simplicial methods for solving a capacited 4-index transportation problem. In: *Journal of numerical analysis and aproximation theory, 2017, vol. 46,* no. 2, p. 181-192. Disponibil: https://ictp.acad.ro/jnaat/journal/article/view/1116/1129.

164. БАСОВА А. В. *Математические модели и генетические методы решения нелинейных задач транспортного типа*: Дис. канд. техн. наук : 05.13.18, 05.13.17 : Ростов н/Д, 2004, 126 с.

165. БЕРЖ, К. *Теория графов и её применения.* Москва: Издательство Иностранной литературы, 1962.

166. ГОЛЬШТЕЙН, Е., ЮДИН, В. Д. *Задачи линейного програмирования транспортного типа.* Москва: Наука, 1969.

167. ДАНТЦИГ, Д. *Линейное программирование его обобщения и применения.* (traducere de АНДРИАНОВА Г. Н., ГОРЬКОВА Л. И., КОРБУТА А. А.) Москва: ПРОГРЕСС, 1966.

168. ДУБРАВИНА Т. В. *Решение модифицированных транспортных задач металлургического комплекса с использованием генетических алгоритмов*: дис. кандидата технических наук : 05.13.01.- Москва, 2005.- 140 с.

169. КАНТОРОВИЧ, Л. В., ГАВУРИН, М. К. Применение математических методов в вопросах анализа грузоперевозок. В *Сборник статей "Проблемы повышения эффективности работы транспорта", АН СССР* , 1949, с. 110-138.

170. ЛОЗОВАНУ, Д. Д. *Экстремально-комбинаторные задачи и алгоритмы их решения.* Кишинев: Штиинца, 1991.

171. Раскин Л. Г., Серая О. В., Дунаевская О. И. Нечеткая модель нелинейной многоиндексной транспортной задачи // ВЕЖПТ. 2012. №4 (60). Disponibil: https://cyberleninka.ru/article/n/nechetkaya-model-nelineynoy-mnogoindeksnoy-transportnoy-zadachi, *[citat: 23.05.2020]*.

172. УПСОН, Р. *Введение в теорию графов.* (traducere de Никитинский Г. П.) Москва: Мир, 1978.

173. ШОР, Н. З. *Методы недифференцируемой оптимизации и сложные экстремальные задачи.* Кишинэу: Эврика, 2008.

ANEXE

Anexa 1

Tratarea soluției degenerate în soluționarea problemei clasice de transport

În cazul în care soluția admisibilă de bază are mai puțin de $m+n-1$ componente pozitive și o soluție admisibilă de bază degenerată se poate obține o soluție nedegenerată înlocuind zerourile în exces cu 0^+ care impune participarea lor în soluție ca valori neegale cu zero.

O altă modalitate (Pașa & Ungureanu, 2017b) constă în alcătuirea sistemului $u_i + v_j = c_{ij}$, $i = \overline{1,m}$, $j = \overline{1,n}$ - bazice. În cazul degenerat numărul de variabile cărora li se pot atribui valori arbitrare este mai mare decât unu. Se soluționează sistemul și se calculează diferențele finite $\delta_{ij} = c_{ij} - (u_i + v_j)$, $i = \overline{1,m}$, $j = \overline{1,n}$. Se alege, celula cu cea mai mică valoare negativă. Pentru ea este găsit un ciclu de redistribuire reieșind din considerentele că toate celule cu etichetă "-" trebuie neapărat să fie ocupate, adică valoarea înscrisă într-o celulă etichetată cu "-" trebuie să fie pozitivă. Existența cel puțin a unui ciclu este garantată de varianta obișnuită a metodei potențialelor. În urma redistribuirii valorilor se efectuează trecerea de la planul degenerat de transport la unul îmbunătățit, dar nu neapărat la unul nedegenerat.

O modalitate de îmbunătățire a soluției sau/și obținerea unei soluții nedegenerate este respectarea următoarelor recomandări:

1. Se construiește un ciclu, care începe cu o celulă ce are valoarea fluxului egală cu zero, cea căreia îi corespunde un cost minim în cazul în care este posibilitatea alegerii din câteva opțiuni și care este marcată cu "+";
2. Pentru ciclu se aleg astfel de celule marcate cu "-" încât neapărat să conțină un flux de produs, iar celulele marcate cu "+" pot fi chiar și toate cu flux nul;
3. Pentru determinarea cantității de produs care trebuie transportat astfel încât să fie îmbunătățită soluția se alege $x_{rs} = min\{x_{ij} | oricare\ i,j\ din\ ciclu\ cu\ semnul\ " - "\}$. După care se modifică variabilele care aparțin ciclului după regula:

$$x_{ij} = \begin{cases} x_{ij} - x_{rs}, & pentru\ celule\ cu\ semnul\ " - " \\ x_{ij} + x_{rs}, & pentru\ celule\ cu\ semnul\ " + " \end{cases}$$

4. Se trece la **pasul 3** a metodei potențialilor pentru a verifica dacă soluția obținută este optimă, în caz contrar se trece la ***punctul 1*** și se repetă procedura de construire a ciclului și îmbunătățirea soluției pentru o soluție degenerată.

O asemenea metodă de îmbunătățire a planurilor de transport degenerate exclude cazurile când se efectuează trecerea de la un plan la altul cu schimbarea doar a valorilor 0^+ dintr-o celulă în alta. Tratarea în așa fel a cazurilor degenerate poate micșora numărul de iterații ale metodei

potențialelor la rezolvarea problemelor degenerate. În cazul respectării recomandărilor descrise mai sus, se ajunge la aceiași soluție optimă care ar fi fost obținută utilizând 0^+ pentru a evita cazul soluției degenerate. Aceste reguli pot conduce la obținerea unei soluții optime în mai puține iterații în cazul unor probleme degenerate.

Exemplu: Fie este dat Tabelul 1.1 de transport cu soluția inițială degenerată a problemei de transport.

Tabelul 1.1. Soluție degenerată a problemei de transport Sursa: elaborat de autor

A_i \ B_j	B_1	B_2	B_3	B_4	a_i
A_1	"-" 9 50	7	2	"+" 1	50
A_2	"+" 4	"-" 10 50	10	2	50
A_3	5	"+" 3	"-" 3 50	7	50
A_4	5	2	"+" 2	"-" 8 50	50
b_j	50	50	50	50	

Din tabel se vede că este obținută o soluție $x_1 = (50, 50, 50{,}50{,}0{,}0{,}0)$ care conține doar 4 componente nenule, deci este o soluție degenerată. În acest caz costul cheltuielilor este $F(x_1) = 1900$. Urmând recomandările de mai sus, se construiește un ciclu (A_1, B_1), (A_2, B_1), (A_2, B_2), (A_3, B_2), (A_3, B_3), (A_4, B_3), (A_4, B_4), (A_1, B_4) și se asociază începând cu (A_1, B_1) semnul "-" iar pentru (A_1, B_4) semnul "+". În urma transportării cantității de produs de mărimea 50 se obține Tabelul 1.2:

Tabelul 1.2. Soluție degenerată îmbunătățită a problemei de transport Sursa: elaborat de autor

A_i \ B_j	B_1	B_2	B_3	B_4	a_i
A_1	9	7	2	1 50	50
A_2	4 50	10	10	2	50
A_3	5	3 50	3	7	50
A_4	5	2	2 50	8	50
b_j	50	50	50	50	

Ca rezultat se obține o soluție degenerată dar căreia îi corespunde un cost mult mai mic $F(x_1) = 700$. Acest proces poate fi repetat fapt care ne conduce la o soluție optimă.

Anexa 2

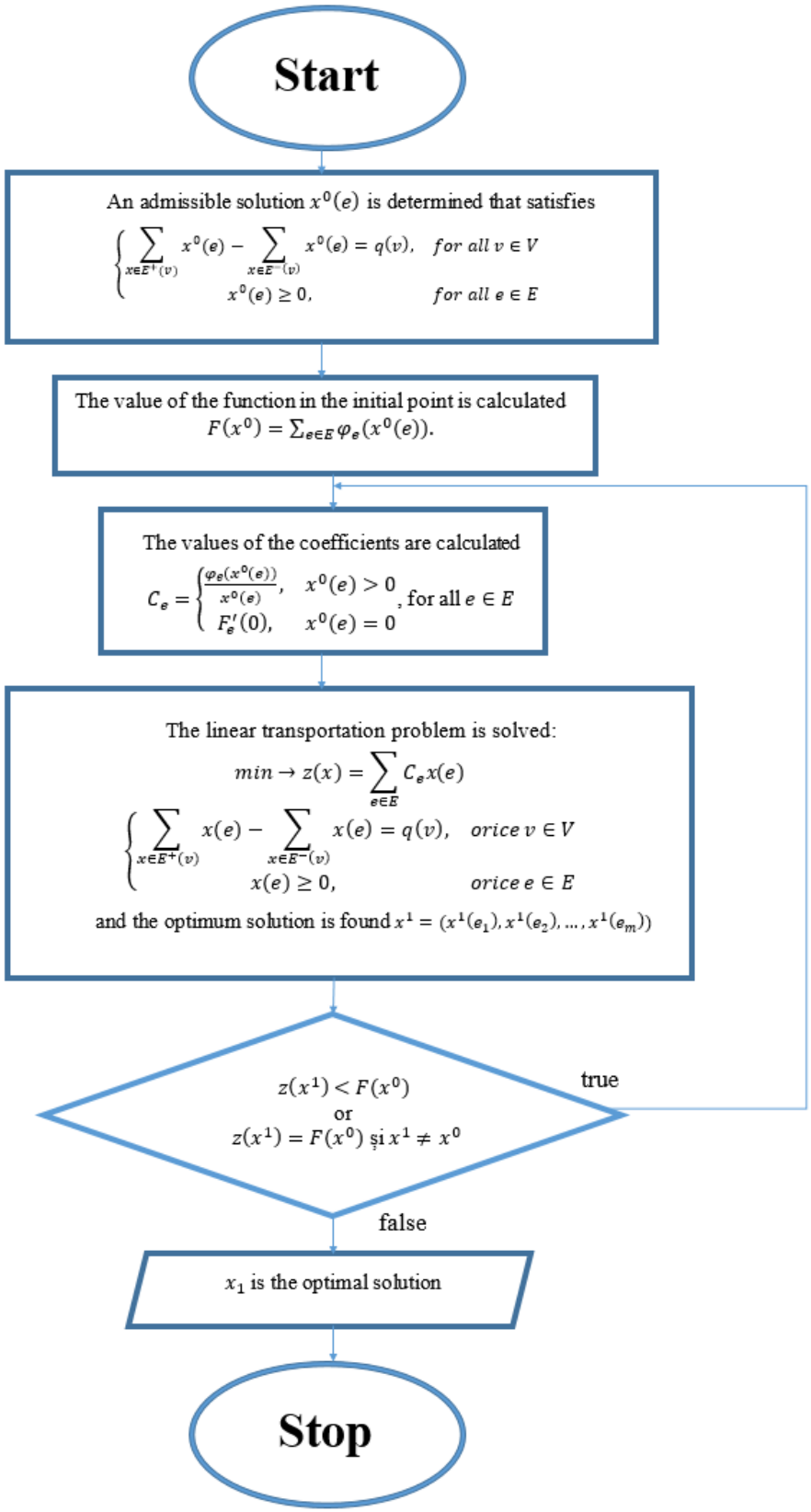
Tabelul 2.1. Algoritmi de soluționare a problemei fluxului maxim Sursa: elaborat de autor

Nr.	*Autori / referințe*	**Anul**	*Complexitate*	*Comentarii*
1	Ford & Fulkerson / [1], [2]	1956	$O(\|E\|f)$	f - flux maxim
3	Dinic / [5]	1970	$O(\|V\|^2\|E\|)$	Fluxuri saturate
2	Edmonds & Karp / [4]	1972	$O(\|V\|\|E\|^2)$	Cel mai scurt drum
4	Karzanov / [6]	1974	$O(\|V\|^3)$	
5	Even & Tarjan / [10]	1975	$O(\|V\|^2\|E\|)$	Prefluxul
6	Galil &Naamad / [7]	1980	$O(\|E\|\|V\|log^2\|V\|)$	Structuri de date
7	Sleator & Tarjan / [8], [24]	1983	$O(\|E\|\|V\|\log\|V\|)$	Arbori binari de căutare
8	Goldberg & Tarjan / [25]	1988	$O(\|E\|\|V\|\log(\frac{\|V\|^2}{\|E\|}))$	Push-Relabel
			$O(\|V\|^2\log\|V\|)$	Implementare paralelă
9	Ahuja & Orlin / [26]	1989	$O(\|V\|\|E\| + \|V\|^2\log\|V\|)$	
			$O(\|V\|^2\log U\log p)$	$p = \lceil\|E\|/\|V\|\rceil$, implementare paralelă
10	Ahuja, Orlin & Tarjan / [27]	1989	$O(\|V\|\|E\|\log\left(\left(\frac{\|V\|}{\|E\|}\right)(\log U)^{\frac{1}{2}}+2\right) + \|E\|\log_{\|V\|}U)$	U - capacitate maximă
11	Miller & Naor / [28]	1995	$O(\sqrt{\|V\|})$	
			$O(I(\|V\|)log^2\|V\|)$	implementare paralelă
12	Goldberg & Rao / [11]	1998	$O(\min\left(\|V\|^{2/3},\sqrt{\|E\|}\right)\|E\|\log(\frac{\|V\|^2}{\|E\|})\log U$	
13	Tarjan, Ward, Zhong, Zhou &Mao / [12]	2006	$O(\|V\|^2\|E\|\log(\|V\|U))$	U - capacitate maximă
14	Borradaile & Klein / [23]	2009	$O(\|V\|\log\|V\|)$	Leftmost
15	Borradaile, Klein, Mazes, Nussbaum, Wulff-Nulsen / [15]	2011	$O(\|V\|log^3\|V\|)$	Pseudoflux

16	Cristiano, Kelner, Madry, Spielman & Teng / [16]	2011	$O(\lvert E\rvert\lvert V\rvert^{1/3}\varepsilon^{-11/3})$	Flux $(1-\varepsilon)-$ aproximativ
17	Kelner, Lee, Orecchia & Sidford / [18]	2014	$O(\lvert\lvert E\rvert\rvert^{1+O(1)}\varepsilon^{-2})$	Flux $\varepsilon-$ aproximativ
18	Madry / [30]	2013	$O\left(\lvert V\rvert^{\frac{10}{7}}\right)=O(\lvert V\rvert^{1.43})$	
19	Orlin / [31]	2013	$(\lvert V\rvert\lvert E\rvert+\lvert E\rvert^{\frac{31}{16}}log\,^{2}\lvert V\rvert)$	
			$O(\lvert V\rvert\lvert E\rvert)$	$\lvert E\rvert=(\lvert V\rvert^{\left(\frac{16}{15}\right)-\varepsilon})$
			$O(\lvert V\rvert^{2}/\log\lvert V\rvert)$	$\lvert E\rvert=O(\lvert V\rvert)$
20	Mehta / [32]	2014	$O(\lvert E\rvert\lvert V\rvert)$	$\lvert E\rvert=O(\lvert V\rvert)$
21	Madry / [33]	2016	$O(\lvert E\rvert^{10/7}U^{1/7})$	

Anexa 3

Schema – bloc a algoritmului AE1 Sursa: elaborat de autor



Anexa 4

Aplicarea algoritmului AE1

Se consideră problema de transport pe rețea cu 5 noduri (o sursă, două destinații și două puncte intermediare) și 7 arce descrisă de graful din Figura 4.1.

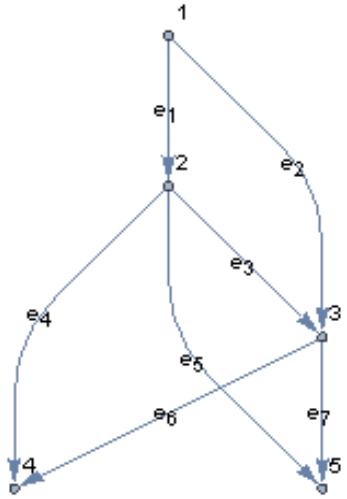


Fig. 4.1. Rețea de transport

Sursa: elaborat de autor

Sursă are disponibilul de flux de 10 u. c., în prima destinație necesarul este de 2 u. c. iar în a doua destinație necesarul este de 8 u. c.

Fiecărui arc $\{e_1, e_2, e_7\}$ îi este asociată funcția de cost:

$$\varphi_1(x) = \begin{cases} x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$$

și arcelor $\{e_3, e_4, e_5, e_6\}$ funcția de cost:

$$\varphi_2(x) = \begin{cases} 2x, & 0 \le x \le 2 \\ 4, & x > 2 \end{cases}.$$

Pasul 1. Se determină o soluție admisibilă a problemei neliniare care satisface restricțiile de existență a fluxului în rețea și se obține $x^0 = (5,5,3,1,1,1,7)$

Iterația 1.

Valoarea funcției obiectiv este $F(x^0) = 13$. Se aproximează problema neliniară cu una liniară și se rezolvă problema liniară pentru care soluția optimă este $x^1 = (2,8,0,2,0,0,8)$ iar funcția obiectiv liniară are valoarea $Z(x^1) = 50/7$. Deoarece $Z(x^1) < F(x^0)$ se substituie x^0 cu x^1 și se trece la *Pasul 2*.

Iterația 2.

Valoarea funcției obiectiv este $F(x^0) = 7$. Se aproximează problema neliniară cu una liniară și se rezolvă problema liniară pentru care soluția optimă este $x^1 = (0,10,0,0,0,2,8)$ iar funcția obiectiv liniară are valoarea $Z(x^1) = 25/4$. Deoarece $Z(x^1) < F(x^0)$ se substituie x^0 cu x^1 și se trece la *Pasul 2*.

Iterația 3.

Valoarea funcției obiectiv este $F(x^0) = 6$. Se aproximează problema neliniară cu una liniară și se rezolvă problema liniară pentru care soluția optimă este $x^1 = (0,10,0,0,0,2,8)$ iar funcția obiectiv liniară are valoarea $Z(x^1) = 6$.

Deoarece $Z(x^1) = F(x^0)$ și $x^1 = x^0$ atunci $x^* = x^1$=(0,10,0,0,0,2,8) este soluție optimă.

Anexa 5

Aplicarea algoritmului AG1

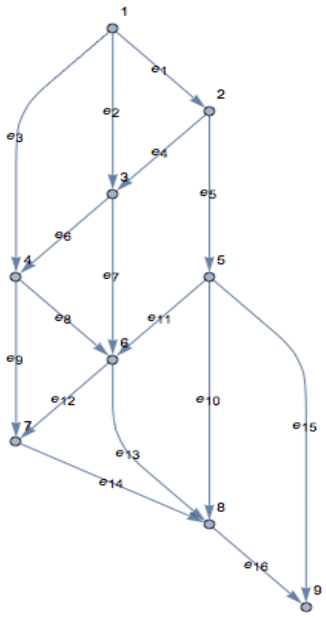


Fig. 5.1. Rețea de transport

Sursa: elaborat de autor

Se consideră rețeaua de transport descrisă de graful din Figura 5.1 cu mulțimea de vârfuri $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ căreia îi este asociată o funcție de producere și consum:

$$q(v) = \begin{cases} 80, & v = 1 \\ 0, & v = 2,3,4,5,6,7,8. \\ -80, & v = 9 \end{cases}$$

Fiecărui arc $\{e_2,\ e_3,\ e_5,\ e_6,\ e_{10},\ e_{11},\ e_{16}\}$ îi este asociată o funcție de cost $\varphi_1(x) = \begin{cases} x, & x \leq 1 \\ 1, & x > 1 \end{cases}$ și arcelor $\{e_1, e_4, e_7, e_8, e_9, e_{12}, e_{13}, e_{14}, e_{15}\}$ funcția de cost $\varphi_2(x) = \begin{cases} 2x, & x \leq 2 \\ 4, & x > 2 \end{cases}$.

Pasul 1. Se generează o populație inițială din 36 de cromozomi:

{{2,2,1,1,3,1,1,1}, {2,1,1,1,1,1,1,1}, {2,2,2,2,2,1,1,1},
{3,2,1,2,3,1,1,1}, {3,1,1,2,3,1,1,1}, {3,2,2,1,3,1,1,1}, {3,2,2,1,1,1,1,1}, {2,1,1,1,1,2,1,1}, {3,2,2,1,2,2,1,1},
{3,1,1,2,2,2,1,1}, {3,2,2,1,2,1,1,1}, {2,1,2,1,3,2,1,1}, {2,2,2,1,2,1,1,1}, {3,2,1,2,3,2,1,1}, {3,1,1,1,3,1,1,1},
{3,2,2,1,3,2,1,1}, {3,1,1,2,1,1,1,1}, {2,2,1,1,2,1,1,1}, {1,2,1,1,2,2,1,1}, {1,2,2,1,1,1,1,1}, {3,2,1,2,1,2,1,1},
{2,1,2,1,3,1,1,1}, {3,2,2,1,3,1,1,1}, {3,2,1,2,3,2,1,1}, {1,2,2,1,3,2,1,1}, {3,2,2,2,3,1,1,1}, {3,2,1,1,2,2,1,1},
{3,1,2,2,3,1,1,1}, {3,1,1,1,1,2,1,1}, {3,1,2,2,2,1,1,1}, {1,2,1,1,1,1,1,1}, {3,2,1,1,3,1,1,1}, {1,2,1,2,3,2,1,1},
{3,2,1,2,3,2,1,1}, {2,1,1,1,1,2,1,1}, {1,1,2,1,2,1,1,1}}

Se inițializează matricea ratelor de flux pentru rețeaua de transport:

{{0.0, 0.6, 0.3, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0},
{0.0, 0.0, 0.1, 0.0, 0.9, 0.0, 0.0, 0.0, 0.0},
{0.0, 0.0, 0.0, 0.6, 0.0, 0.4, 0.0, 0.0, 0.0},
{0.0, 0.0, 0.0, 0.0, 0.0, 0.2, 0.8, 0.0, 0.0},
{0.0, 0.0, 0.0, 0.0, 0.0, 0.3, 0.0, 0.3, 0.4},
{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.7, 0.3, 0.0},
{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0},
{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0},
{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0}}.

Valoarea funcției obiectiv calculată pentru soluția asociată primului cromozom generat aliator este $F(x) = 10$ u.c. și valoarea $F_T(x)$= 546.6 u.c..

Iterația 1, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 18 cromozomi sunt transferați în următoarea populație.

Cromozomii-părinți:

{{4.0,{1,1,2,1,2,1,1,1}},
{4.0,{1,2,1,1,1,1,1,1}},
{4.0,{1,2,1,1,2,2,1,1}},

Cromozomii-urmași obținuți după încrucișare:

{{1,1,2,1,1,1,1,1},
{1,2,1,1,2,1,1,1},
{1,2,1,1,2,1,1,1},

{4.0,{1,2,2,1,1,1,1,1}},	{1,2,2,1,1,2,1,1},
{4.0,{1,2,2,1,3,2,1,1}},	{1,2,2,1,3,2,1,1},
{10.0,{1,2,1,2,3,2,1,1}},	{1,2,1,2,3,2,1,1},
{10.0,{2,1,1,1,1,1,1,1}},	{2,1,1,1,1,1,1,1},
{10.0,{2,2,1,1,2,1,1,1}},	{2,2,1,1,2,1,1,1},
{10.0,{2,2,1,1,3,1,1,1}},	{2,2,1,1,3,1,1,1},
{11.0,{2,1,1,1,1,2,1,1}},	{2,1,1,1,1,2,1,1},
{11.0,{2,1,1,1,1,2,1,1}},	{2,1,1,1,3,1,1,1},
{12.0,{3,1,1,1,3,1,1,1}},	{3,1,1,1,1,2,1,1},
{13.0,{3,1,1,1,1,2,1,1}},	{3,1,1,1,1,1,1,1},
{13.0,{3,1,1,2,1,1,1,1}},	{3,1,1,2,1,2,1,1},
{13.0,{3,1,1,2,3,1,1,1}},	{3,1,1,1,3,1,1,1},
{14.0,{2,1,2,1,3,1,1,1}},	{2,1,2,2,3,1,1,1},
{14.0,{2,2,2,1,2,1,1,1}},	{2,2,2,1,2,1,1,1},
{14.0,{3,1,1,2,2,2,1,1}}}	{3,1,1,2,2,2,1,1}}

Asupra cromozomilor-urmași este aplicată mutația cu o probabilitate 0,01. Fiecărui cromozom i se asociază o soluție în care este calculată valoarea funcției obiectiv.

Populația nouă după mutații:	Soluțiile asociate cromozomilor:
{{4.0,{1,1,2,1,2,1,1,1}},	{{4.0,{0,0,80.0,0.,0,0.,0.,0,80.0,0.,0,0.,0.,0,80.0,80.0}},
{4.0,{1,2,1,1,1,1,1,1}},	{4.0,{0,0,80.0,0.,0.,0,0.,0,80.0,0,0,0.,0.,0,80.0,80.0}},
{4.0,{1,2,1,1,2,2,1,1}},	{4.0,{0,0,80.0,0.,0.,0,0.,0,80.0,0.,0,0.,0.,0.,80.0,80.0}},
{4.0,{1,2,2,1,1,1,1,1}},	{4.0,{0,0,80.0,0.,0.,0.,0.,0,80.0,0,0,0,0.,0.,0,80.0,80.0}},
{4.0,{1,2,2,1,3,2,1,1}},	{4.0,{0,0,80.0,0.,0.,0.,0.,0,80.0,0,0.,0.,0.,0.,0.,80.0,80.0}},
{10.0,{1,2,1,2,3,2,1,1}},	{10.0,{0,0,80.0,0.,0.,0,0.,62.5,17.4,0.,0.,0.,42.7,19.8,60.2,80.0}},
{10.0,{2,1,1,1,1,1,1,1}},	{10.0,{0,30.4,49.6,0.,0,0,30.4,0,49.6,0,0,0.,30.4,0,80.0,80.0}},
{10.0,{2,2,1,1,2,1,1,1}},	{10.0,{0,30.4,49.6,0.,0.,0,30.4,0,49.6,0.,0,0.,30.4,0,80.0,80.0}},
{10.0,{2,2,1,1,3,1,1,1}},	{10.0,{0,30.4,49.6,0.,0.,0,30.4,0,49.6,0.,0.,0.,30.4,0,80.0,80.0}},
{11.0,{2,1,1,1,1,2,1,1}},	{11.0,{0,30.4,49.6,0.,0,0,30.4,0,49.6,0,0,0.,20.8,9.6,70.4,80.0}},
{11.0,{2,1,1,1,1,2,1,1}},	{11.0,{0,30.4,49.6,0.,0,0,30.4,0,49.6,0,0,0.,20.8,9.6,70.4,80.0}},
{12.0,{3,1,1,1,3,1,1,1}},	{12.0,{6.0,3.1,45.8,6.0,0,0,34.2,0,45.8,0.,0.,0.,34.2,0,80.0,80.0}},
{13.0,{3,1,1,1,1,2,1,1}},	{13.0,{6.0,28.1,45.8,6.0,0,0,34.2,0,45.8,0,0,0.,23.3,10.8,69.2,80.0}},
{13.0,{3,1,1,2,1,1,1,1}},	{13.0,{6.0,28.1,45.8,6.0,0,0,34.2,35.8,10.0,0,0,0.,70.0,0,80.0,80.0}},
{13.0,{3,1,1,2,3,1,1,1}},	{13.0,{6.0,28.1,45.8,6.0,0,0,34.2,35.8,10.0,0.,0.,0.,70.0,0,80.0,80.0}},
{14.0,{2,1,2,1,3,1,1,1}},	{14.0,{0,30.4,49.6,0.,0,13.4,17.0,0,63.0,0.,0.,0.,17.0,0,80.0,80.0}},
{14.0,{2,2,2,1,2,1,1,1}},	{14.0,{0,30.4,49.6,0.,0.,13.4,17.0,0,63.0,0.,0,0.,17.0,0,80.0,80.0}},
{14.0,{3,1,1,2,2,2,1,1}},	{14.0,{6.0,28.1,45.8,6.0,0,0,34.2,35.8,10.0,0.,0,0.,47.8,22.2,57.8,80.0}},
{4.0,{1,1,2,1,1,1,1,1}},	{4.0,{0,0,80.0,0.,0,0.,0.,0,80.0,0,0,0.,0.,0,80.0,80.0}},
{4.0,{1,2,1,1,2,1,1,1}},	{4.0,{0,0,80.0,0.,0.,0,0.,0,80.0,0.,0,0.,0.,0,80.0,80.0}},
{4.0,{1,2,1,1,2,1,1,1}},	{4.0,{0,0,80.0,0.,0.,0,0.,0,80.0,0,0,0.,0.,0,80.0,80.0}},
{4.0,{1,2,2,1,1,2,1,1}},	{4.0,{0,0,80.0,0.,0.,0.,0.,0,80.0,0,0,0,0.,0.,0,80.0,80.0}},
{4.0,{1,2,2,1,3,2,1,1}},	{4.0,{0,0,80.0,0.,0.,0.,0.,0,80.0,0.,0.,0.,0.,0.,80.0,80.0}},
{10.0,{1,2,1,2,3,2,1,1}},	{10.0,{0,0,80.0,0.,0.,0,0.,62.5,17.4,0.,0.,0.,42.7,19.8,60.2,80.0}},
{10.0,{2,1,1,1,1,1,1,1}},	{10.0,{0,30.4,49.6,0.,0,0,30.4,0,49.6,0,0,0.,30.4,0,80.0,80.0}},
{10.0,{2,2,1,1,2,1,1,1}},	{10.0,{0,30.4,49.6,0.,0.,0,30.4,0,49.6,0.,0,0.,30.4,0,80.0,80.0}},
{10.0,{2,2,1,1,3,1,1,1}},	{10.0,{0,30.4,49.6,0.,0.,0,30.4,0,49.6,0.,0.,0.,30.4,0,80.0,80.0}},
{11.0,{2,1,1,1,1,2,1,1}},	{11.0,{0,30.4,49.6,0.,0,0,30.4,0,49.6,0,0,0.,20.8,9.6,70.4,80.0}},
{10.0,{2,1,1,1,3,1,1,1}},	{10.0,{0,30.4,49.6,0.,0,0,30.4,0,49.6,0.,0.,0.,30.4,0,80.0,80.0}},
{13.0,{3,1,1,1,1,2,1,1}},	{13.0,{6.0,28.1,45.8,6.0,0,0,34.2,0,45.8,0,0,0.,23.3,10.8,69.2,80.0}},
{12.0,{3,1,1,1,1,1,1,1}},	{12.0,{6.0,28.1,45.8,6.0,0,0,34.2,0,45.8,0,0,0.,34.2,0,80.0,80.0}},
{14.0,{3,1,1,2,1,2,1,1}},	{14.0,{6.0,28.1,45.8,6.0,0,0,34.2,35.8,10.0,0,0,0.,47.8,22.2,57.8,80.0}},
{12.0,{3,1,1,1,3,1,1,1}},	{12.0,{6.0,28.1,45.8,6.0,0,0,34.2,0,45.8,0.,0.,0.,34.2,0,80.0,80.0}},
{15.0,{2,1,2,2,3,1,1,1}},	{15.0,{0,30.4,49.6,0.,0,13.4,17.0,49.3,13.7,0.,0.,0.,66.2,0,80.0,80.0}},
{14.0,{2,2,2,1,2,1,1,1}},	{14.0,{0,30.4,49.6,0.,0.,0,13.4,17.0,0,63.0,0.,0,0.,17.0,0,80.0,80.0}},
{14.0,{3,1,1,2,2,2,1,1}}}	{14.0,{6.0,28.1,45.8,6.0,0,0,34.2,35.8,10.0,0.,0,0.,47.8,22.2,57.8,80.0}}}

Valoarea minimă a funcției obiectiv pentru populația generată este $F(x) = 4$ u.c.. Valoarea $F_T(x)$= 350 u.c. ceea ce presupune o micșorare a costului total per populația nouă în comparație cu cea precedentă.

Iterația 2, se execută succesiv pașii 2 – 6. Primii 18 cromozomi sunt transferați în următoarea populație. Asupra cromozomilor-urmași este aplicată mutație cu o probabilitate 0,01. Fiecărui cromozom i se asociază o soluție în care este calculată valoarea funcției obiectiv. Valoarea minimă a funcției obiectiv pentru populația generată este $F(x) = 4$ u.c.. Deoarece această mărime este egală cu mărimea minimă a funcției obiectiv la pasul precedent, atunci aceasta este soluția optimă a problemei formulate x={0,80.,0,0,0.,80.,0,0,80.,0.,0,0,0.,0,80.,80.}. STOP.

Anexa 6

Aplicarea algoritmului AG2

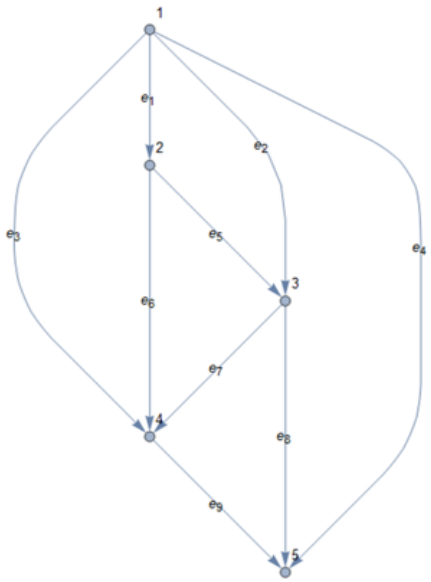


Fig. 6.1. Rețea de transport

Sursa: elaborat de autor

Se consideră rețeaua de transport descrisă de graful din Figura 6.1 cu mulțimea de vârfuri {1,2,3,4,5} căreia îi este asociată o funcție de producere și consum:

$$q(v) = \begin{cases} 15, & v = 1 \\ 0, & v = 2,3,4 \\ -15, & v = 5 \end{cases}$$

Fiecărui arc din mulțimea de arce $\{e_1, e_3, e_4, e_7\}$ îi este asociată o funcție de cost $\varphi_1(x) = \begin{cases} x, & x \leq 1 \\ 1, & x > 1 \end{cases}$ și $\varphi_2(x) = \begin{cases} 2x, & x \leq 2 \\ 4, & x > 2 \end{cases}$ - arcelor $\{e_2, e_5, e_6, e_8, e_9\}$.

Pasul 1. Se generează o populație inițială din 20 de cromozomi și se calculează valoarea funcției obiectiv asociată fiecărui cromozom din populație:

{{17.,{{0.39,0.01,0.23,0.37},{0.48,0.52},{0.18,0.82},{1.}}},
{16.,{{0.04,0.37,0.33,0.26},{0.66,0.34},{0.6,0.4},{1.}}},
{18.,{{0.33,0.19,0.15,0.33},{0.33,0.67},{0.51,0.49},{1.}}},
{14.,{{0.01,0.24,0.23,0.52},{0.27,0.73},{0.19,0.81},{1.}}},
{18.,{{0.25,0.2,0.38,0.17},{0.7,0.3},{0.81,0.19},{1.}}},
{18.,{{0.34,0.08,0.48,0.10},{0.59,0.41},{0.54,0.46},{1.}}},
{17.,{{0.23,0.28,0.16,0.33},{0.61,0.39},{0.5,0.5},{1.}}},
{16.,{{0.19,0.32,0.11,0.38},{0.81,0.19},{0.67,0.33},{1.}}},
{18.,{{0.08,0.23,0.56,0.13},{0.41,0.59},{0.62,0.38},{1.}}},
{18.,{{0.61,0.14,0.23,0.024},{0.4,0.6},{0.32,0.68},{1.}}},
{17.,{{0.12,0.012,0.65,0.22},{0.13,0.87},{0.45,0.55},{1.}}},
{17.,{{0.44,0.08,0.44,0.04},{0.18,0.82},{0.1,0.9},{1.}}},
{19.,{{0.21,0.35,0.27,0.18},{0.34,0.66},{0.82,0.18},{1.}}},
{16.,{{0.28,0.37,0.11,0.25},{0.22,0.78},{0.81,0.19},{1.}}},
{19,{{0.28,0.17,0.28,0.27},{0.32,0.68},{0.76,0.24},{1.}}},
{13.,{{0.03,0.15,0.73,0.09},{0.89,0.11},{0.03,0.97},{1.}}},
{19.,{{0.23,0.30,0.42,0.05},{0.41,0.59},{0.54,0.46},{1.}}},
{19.,{{0.12,0.2,0.33,0.35},{0.25,0.75},{0.6,0.4},{1.}}},
{16.,{{0.23,0.29,0.26,0.22},{0.9,0.1},{0.6,0.4},{1.}}},
{15.,{{0.16,0.35,0.07,0.42},{0.34,0.66},{0.28,0.72},{1.}}}}.

Valoarea funcției obiectiv calculată pentru soluția asociată primului cromozom generat aleator este $F(x) = 17$ u.c. și valoarea $F_T(x) = 340$ u.c..

Iterația 1, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 10 cromozomi sunt transferați în următoarea populație.

Cromozomii-părinții:	Cromozomii-urmași după încrucișare
13.,{{0.03,0.15,0.73,0.09},{0.89,0.11},{0.03,0.97},{1.}}	{0.03,0.15,0.73,0.09},{0.27,0.73},{0.19,0.81},{1.}
14.,{{0.02,0.24,0.23,0.52},{0.27,0.73},{0.19,0.81},{1.}}	{0.02,0.24,0.23,0.52},{0.89,0.11},{0.03,0.97},{1.}
15.,{{0.16,0.35,0.07,0.42},{0.34,0.66},{0.28,0.72},{1.}}	{0.16,0.35,0.07,0.42},{0.81,0.19},{0.67,0.33},{1.}
16.,{{0.19,0.32,0.11,0.38},{0.81,0.19},{0.67,0.33},{1.}}	{0.19,0.32,0.11,0.38},{0.34,0.66},{0.28,0.72},{1.}
16.,{{0.05,0.37,0.33,0.26},{0.66,0.34},{0.6,0.4},{1.}}	{0.05,0.37,0.33,0.26},{0.66,0.34},{0.6,0.4},{1.}
16.,{{0.23,0.29,0.26,0.22},{0.9,0.1},{0.6,0.4},{1.}}	{0.23,0.29,0.26,0.22},{0.9,0.1},{0.6,0.4},{1.}
16.,{{0.28,0.37,0.11,0.25},{0.22,0.78},{0.81,0.19},{1.}}	{0.28,0.37,0.11,0.25},{0.22,0.78},{0.18,0.82},{1.}
17.,{{0.39,0.01,0.23,0.38},{0.48,0.52},{0.18,0.82},{1.}}	{0.39,0.01,0.23,0.37},{0.48,0.52},{0.81,0.19},{1.}
17.,{{0.12,0.01,0.65,0.22},{0.13,0.87},{0.45,0.55},{1.}}	{0.12,0.01,0.65,0.22},{0.61,0.39},{0.5,0.5},{1.}
17.,{{0.23,0.28,0.16,0.33},{0.61,0.39},{0.5,0.5},{1.}}	{0.23,0.28,0.16,0.33},{0.13,0.87},{0.45,0.55},{1.}

Asupra cromozomilor-urmași este aplicată mutația cu o probabilitate 0,01. Fiecărui cromozom i se asociază o soluție în care este calculată valoarea funcției obiectiv.

Populația noua după mutații:	Soluțiile asociate cromozomilor
13.,{{0.03,0.15,0.73,0.09},{0.89,0.11},{0.03,0.97},{1.}}	13.,{11.,0.44,1.3,2.3,9.8,1.2,10.,0.27,11.}
14.,{{0.01,0.24,0.23,0.52},{0.27,0.73},{0.19,0.81},{1.}}	14.,{3.4,0.27,7.7,3.6,0.92,2.5,0.96,0.22,8.7}
15.,{{0.16,0.35,0.07,0.42},{0.34,0.66},{0.28,0.72},{1.}}	15.,{1.,2.4,6.3,5.3,0.35,0.68,2.,0.74,8.3}
16.,{{0.19,0.32,0.11,0.38},{0.81,0.19},{0.67,0.33},{1.}}	16.,{1.6,2.9,5.7,4.9,1.3,0.31,1.4,2.8,7.}
16.,{{0.04,0.37,0.33,0.26},{0.66,0.34},{0.6,0.4},{1.}}	16.,{5.,0.67,3.8,5.5,3.3,1.7,1.6,2.4,5.4}
16.,{{0.23,0.29,0.26,0.22},{0.9,0.1},{0.6,0.4},{1.}}	16.,{3.9,3.4,3.3,4.3,3.5,0.39,2.8,4.1,6.1}
16.,{{0.27,0.37,0.11,0.25},{0.22,0.78},{0.81,0.19},{1.}}	16.,{1.6,4.1,3.7,5.5,0.36,1.3,0.87,3.6,4.6}
17.,{{0.39,0.01,0.23,0.37},{0.48,0.52},{0.18,0.82},{1.}}	17.,{3.4,5.9,5.7,0.11,1.6,1.8,6.1,1.3,12.}
17.,{{0.12,0.01,0.65,0.22},{0.13,0.87},{0.45,0.55},{1.}}	17.,{9.7,1.8,3.3,0.18,1.3,8.4,1.7,1.4,5.}
17.,{{0.23,0.28,0.16,0.33},{0.61,0.39},{0.5,0.5},{1.}}	17.,{2.4,3.5,5.,4.1,1.5,0.93,2.5,2.5,7.5}
15.,{{0.03,0.15,0.73,0.09},{0.27,0.73},{0.19,0.81},{1.}}	15.,{11.,0.44,1.3,2.3,3.,8.,2.8,0.64,4.}
10.,{{0.02,0.24,0.23,0.52},{0.89,0.11},{0.03,0.97},{1.}}	10.,{3.4,0.27,7.7,3.6,3.1,0.37,3.2,0.09,11.}
15.,{{0.16,0.35,0.07,0.42},{0.81,0.19},{0.67,0.33},{1.}}	15.,{1.,2.4,6.3,5.3,0.83,0.2,1.1,2.1,7.4}
16.,{{0.19,0.32,0.11,0.38},{0.34,0.66},{0.28,0.72},{1.}}	16.,{1.6,2.9,5.7,4.9,0.54,1.1,2.5,0.94,8.1}
16.,{{0.04,0.37,0.33,0.26},{0.66,0.34},{0.6,0.4},{1.}}	16.,{5.,0.67,3.8,5.5,3.3,1.7,1.6,2.3,5.4}
16.,{{0.23,0.29,0.26,0.22},{0.9,0.1},{0.6,0.4},{1.}}	16.,{3.9,3.4,3.3,4.3,3.5,0.39,2.8,4.2,6.1}
16.,{{0.28,0.37,0.11,0.25},{0.22,0.78},{0.18,0.82},{1.}}	16.,{1.6,4.1,3.7,5.5,0.36,1.3,3.7,0.81,7.4}
17.,{{0.39,0.01,0.23,0.37},{0.48,0.52},{0.81,0.19},{1.}}	17.,{3.4,5.9,5.7,0.11,1.6,1.8,1.4,6.,7.1}
17.,{{0.12,0.012,0.65,0.22},{0.61,0.39},{0.5,0.5},{1.}}	17.,{9.7,1.8,3.3,0.18,6.,3.7,3.9,3.9,7.2}
18.,{{0.23,0.28,0.16,0.33},{0.13,0.87},{0.45,0.55},{1.}}	18.,{2.4,3.5,5.,4.1,0.32,2.1,2.1,1.7,7.1}

Valoarea minimă a funcției obiectiv este F(x)=10 u.c.. Valoarea $F_T(x)$= 310 u.c. ceea ce presupune o micșorare a costului total per populația nouă în comparație cu cea inițială.

Iterația 2, se execută succesiv pașii 2 – 6. Primii 10 cromozomi sunt transferați în următoarea populație. Asupra cromozomilor-urmași este aplicată mutația cu o probabilitate 0,01. Fiecărui cromozom i se asociază o soluție și este calculată valoarea funcției obiectiv pentru fiecare soluție obținută. Valoarea $F_T(x)$= 290 u.c. ceea ce presupune o micșorare a costului total per populația nouă în comparație cu cea precedentă. Valoarea minimă a funcției obiectiv pentru populația generată este F(x)=10 u.c.. Deoarece această mărime este egală cu valoarea minimă a funcției obiectiv la pasul precedent, aceasta este considerată soluția optimă a problemei formulate x={3.4,0.27,7.7,3.6,3.1,0.37,3.2,0.09,11.}. STOP.

Anexa 7

Aplicarea algoritmului AG3

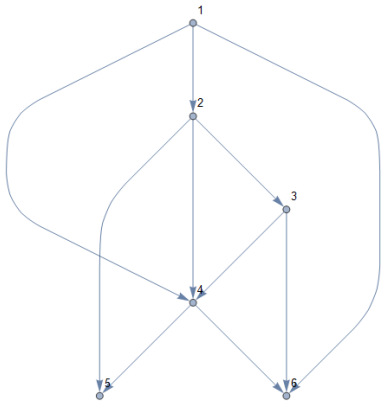


Fig. 7.1. Rețea de transport cu o sursă și 2 destinații
Sursa: elaborat de autor

Se consideră rețeaua de transport descrisă de graful din Figura 7.1. Mulțimii de vârfuri $\{1,2,3,4,5,6\}$ îi este asociată o funcție de producere și consum de forma:

$$q(v) = \begin{cases} 100, & v = 1 \\ 0, & v = 2,3,4 \\ -27, & v = 5 \\ -73, & v = 6. \end{cases}$$

Fiecărui arc $\{e_1, e_2, e_3, \dots, e_{10}\}$ îi este asociată o funcție de cost $\varphi_1(x) = \begin{cases} x, & x \leq 1 \\ 1, & x > 1 \end{cases}$ și $\varphi_2(x) = \begin{cases} 2x, & x \leq 2 \\ 4, & x > 2 \end{cases}$. Se cere determinarea unui astfel de flux încât transportarea a 100 u. c. de flux din sursa $\{1\}$ în destinațiile $\{5,6\}$ să fie de cost minim cu restricția că fluxul care pornește din sursă să satisfacă în totalitate necesitățile destinațiilor. Asupra cromozomilor-urmași este aplicată mutația cu o probabilitate 0,01. Soluția optimă a problemei este de forma $x = \{x_1, x_2, \dots, x_{10}\}$.

Pasul 1. Se generează o populație inițială din 24 cromozomi din care sunt șterși cei ce se repetă. Astfel, populația cu care se operează conține 13 cromozomi diferiți. Se determină soluțiile asociate cromozomilor și se calculează valoarea funcției obiectiv pentru fiecare soluție:

{{1→2,2→3,1→4,2→5,4→6},{1→2,2→3,3→4,2→5,3→6},{1→2,2→3,3→4,2→5,4→6},
{1→2,2→3,1→4,4→5,4→6},{1→2,2→3,1→4,2→5,1→6},{1→2,2→3,2→4,4→5,4→6},
{1→2,2→3,1→4,2→5,3→6},{1→2,2→3,3→4,4→5,1→6},{1→2,2→3,2→4,2→5,3→6},
{1→2,2→3,2→4,2→5,1→6},{1→2,2→3,2→4,2→5,4→6},{1→2,2→3,2→4,4→5,3→6},
{1→2,2→3,3→4,4→5,3→6}}.

Iterația 1, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 6 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Populația nouă care constă din cromozomi-părinți și cromozomi-urmași este:

{{1→2,2→3,1→4,4→5,4→6},{1→2,2→3,1→4,2→5,1→6},{1→2,2→3,2→4,2→5,1→6},
{1→2,2→3,1→4,2→5,4→6},{1→2,2→3,3→4,2→5,3→6},{1→2,2→3,2→4,4→5,4→6},
{1→2,1→6,2→3,2→4,4→5},{1→2,1→4,2→3,2→5,4→6},{1→2,1→4,2→3,2→5,4→6},
{1→2,1→4,2→3,2→5,4→6},{1→2,2→3,2→4,2→5,4→6},{1→2,2→3,2→4,3→6,4→5}}.

Valoarea funcției obiectiv totală a populației este $F_T(x) = 348$ u.c. iar valoarea funcției obiectiv $F(x) = 11$ u.c.. Soluția $x = \{80, 100, 0, 0, 0, 0, 0, 0, 27, 73\}$ este asociată cromozomului {1→2,2→3,1→4,4→5,4→6}.

Iterația 2, se execută succesiv pașii 2 – 6 . Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 6 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Valoarea $F_T(x) = 259$ u.c. iar valoarea funcției obiectiv pentru primul cromozom din populația obținută este $F(x) = 10$ u.c.. Soluția x=(0, 100, 0, 0, 0, 0, 0, 0, 27, 73) este asociată cromozomului {1→2,2→3,1→4,4→5,4→6} cu cele mai bune caracteristici.

Iterația 3, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 6 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Valoarea $F_T(x) = 250$ u.c. iar valoarea funcției obiectiv pentru primul cromozom din populația obținută este $F(x) = 10$ u.c.. Soluția x=(0, 27, 73, 0, 0, 0, 0, 0, 27, 0) este asociată cromozomului {1→2,1→4,1→6,2→3,4→5} cu cele mai bune caracteristici care este soluția problemei după executarea a 3 iterații. STOP.

Anexa 8

Aplicarea algoritmului AG4

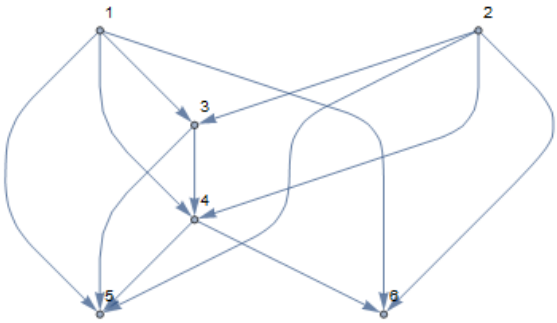


Fig. 8.1. Rețea de transport cu o sursă și 2 destinații Sursa: elaborat de autor

Se consideră rețeaua de transport descrisă de graful din Figura 8.1. Mulțimii de vârfuri {1,2,3,4,5,6} îi este asociată o funcție de producere și consum de forma:

$$q(v) = \begin{cases} 29, & v = 1 \\ 21, & v = 2 \\ 0, & v = 3,4 \\ -9, & v = 5 \\ -41, & v = 6. \end{cases}$$

Fiecărui arc $\{e_2, e_6, e_8, e_9, e_{11}\}$ îi este asociată o funcție de cost $\varphi_1(x) = \begin{cases} x, & x \leq 1 \\ 1, & x > 1 \end{cases}$ și arcelor $\{e_1, e_3, e_4, e_5, e_7, e_{10}, e_{12}\}$ funcția de cost $\varphi_2(x) = \begin{cases} 2x, & x \leq 2 \\ 4, & x > 2 \end{cases}$, unde $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}\}$ sunt arcele {1→3,1→4,1→5,1→6,2→3,2→4,2→5,2→6,3→4,3→5,4→5,4→6}. Asupra cromozomilor-urmași este aplicată mutația cu o probabilitate 0,01.

Se cere determinarea unui astfel de flux încât transportarea a 50 u. c. de flux din sursele {1,2} în destinațiile {5,6} să fie de cost minim cu restricția că fluxul ce pornește din surse satisface în totalitate destinațiile. Soluția problemei este de forma $x = \{x_1, x_2, \dots, x_{12}\}$.

Pasul 1. Se generează o populație inițială din 24 cromozomi:

{{{1→3,3→4,3→5,4→6},{2→3,2→4,4→5,2→6}},{1,2},{1,2}},
{{1→3,3→4,4→5,4→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1}},
{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,4→6}},{1,2},{2,1}},
{{1→3,1→4,4→5,1→6},{2→3,2→4,3→5,2→6}},{2,1},{1,2}},
{{1→3,3→4,3→5,1→6},{2→3,2→4,4→5,4→6}},{2,1},{1,2}},
{{1→3,3→4,3→5,4→6},{2→3,3→4,2→5,4→6}},{2,1},{2,1}},
{{1→3,1→4,4→5,1→6},{2→3,2→4,4→5,4→6}},{1,2},{2,1}},
{{1→3,1→4,1→5,1→6},{2→3,3→4,3→5,4→6}},{2,1},{2,1}},
{{1→3,3→4,4→5,1→6},{2→3,3→4,2→5,4→6}},{1,2},{2,1}},
{{1→3,3→4,1→5,4→6},{2→3,2→4,2→5,4→6}},{2,1},{2,1}},
{{1→3,3→4,4→5,4→6},{2→3,3→4,3→5,2→6}},{1,2},{2,1}},
{{1→3,3→4,1→5,4→6},{2→3,2→4,2→5,4→6}},{1,2},{2,1}},
{{1→3,3→4,4→5,4→6},{2→3,2→4,3→5,2→6}},{2,1},{2,1}},
{{1→3,3→4,3→5,1→6},{2→3,3→4,4→5,4→6}},{1,2},{2,1}},
{{1→3,1→4,1→5,4→6},{2→3,3→4,3→5,4→6}},{1,2},{1,2}},
{{1→3,3→4,3→5,4→6},{2→3,2→4,2→5,2→6}},{1,2},{2,1}},
{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,4→6}},{2,1},{1,2}},
{{1→3,1→4,4→5,4→6},{2→3,2→4,3→5,2→6}},{2,1},{1,2}},
{{1→3,3→4,3→5,4→6},{2→3,3→4,4→5,4→6}},{2,1},{2,1}},
{{1→3,1→4,1→5,1→6},{2→3,2→4,2→5,2→6}},{2,1},{1,2}},
{{1→3,1→4,3→5,1→6},{2→3,2→4,2→5,2→6}},{2,1},{2,1}},
{{1→3,3→4,4→5,4→6},{2→3,3→4,2→5,4→6}},{2,1},{2,1}},
{{1→3,3→4,3→5,4→6},{2→3,2→4,4→5,2→6}},{2,1},{1,2}},

{{1→3,3→4,3→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{2,1}}}

Iterația 1, se execută succesiv pașii 2 – 6.

Cromozomii-părinți care participă la încrucișare:	Cromozomii-urmași obținuți după încrucișare:
{{9,{{1→3,1→4,1→5,1→6},{2→3,2→4,2→5,2→6}},{2,1},{1,2}},	{{1→3,1→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{2,1},{2,1}},
{9,{{1→3,3→4,3→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{2,1}},	{{1→3,3→4,3→5,1→6},{2→3,2→4,2→5,2→6}},{2,1},{1,2}},
{10,{{1→3,1→4,4→5,1→6},{2→3,2→4,4→5,4→6}},{1,2},{2,1}},	{{1→3,1→4,4→5,1→6},{2→3,2→4,4→5,4→6}},{1,2},{2,1}},
{10,{{1→3,3→4,3→5,1→6},{2→3,2→4,4→5,4→6}},{2,1},{1,2}},	{{1→3,3→4,3→5,1→6},{2→3,2→4,4→5,4→6}},{2,1},{1,2}},
{11,{{1→3,3→4,4→5,4→6},{2→3,2→4,3→5,2→6}},{2,1},{2,1}},	{{1→3,3→4,4→5,4→6},{2→3,2→4,4→5,2→6}},{1,2},{1,2}},
{12,{{1→3,3→4,3→5,4→6},{2→3,2→4,4→5,2→6}},{2,1},{1,2}},	{{1→3,3→4,3→5,4→6},{2→3,2→4,3→5,2→6}},{2,1},{2,1}},
{13,{{1→3,1→4,3→5,1→6},{2→3,2→4,2→5,2→6}},{2,1},{2,1}},	{{1→3,1→4,3→5,1→6},{2→3,2→4,3→5,2→6}},{2,1},{1,2}},
{13,{{1→3,1→4,4→5,1→6},{2→3,2→4,3→5,2→6}},{2,1},{1,2}},	{{1→3,1→4,4→5,1→6},{2→3,2→4,2→5,2→6}},{2,1},{1,2}},
{14,{{1→3,1→4,1→5,4→6},{2→3,3→4,3→5,4→6}},{1,2},{1,2}},	{{1→3,1→4,1→5,4→6},{2→3,2→4,3→5,2→6}},{2,1},{2,1}},
{14,{{1→3,1→4,4→5,4→6},{2→3,2→4,3→5,2→6}},{2,1},{1,2}},	{{1→3,1→4,4→5,4→6},{2→3,3→4,3→5,4→6}},{2,1},{1,2}},
{14,{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,4→6}},{1,2},{2,1}},	{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,4→6}},{1,2},{1,2}},
{14,{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,4→6}},{2,1},{1,2}},	{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,4→6}},{1,2},{2,1}}}

Valoarea $F_T(x) = 283$ u.c. iar valoarea funcției obiectiv pentru primul cromozom din populația obținută este $F(x) = 9$ u.c.. Soluția x=(0, 0, 0, 29, 0, 0, 9, 12, 0, 0, 0, 0) este asociată cromozomului {{1→3,1→4,1→5,1→6},{2→3,2→4,2→5,2→6},{2,1},{1,2}} cu cele mai bune caracteristici.

Iterația 2, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 12 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Valoarea $F_T(x) = 251$ u.c. iar valoarea funcției obiectiv pentru primul cromozom din populația obținută este $F(x) = 9$ u.c.. Soluția x=(0, 0, 0, 29, 0, 0, 9, 12, 0, 0, 0, 0) este asociată cromozomului {{1→3,1→4,1→5,1→6},{2→3,2→4,2→5,2→6},{2,1},{1,2}} cu cele mai bune caracteristici.

Iterația 3, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 12 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Valoarea $F_T(x) = 233$ u.c. iar valoarea funcției obiectiv pentru primul cromozom din populația obținută este $F(x) = 7$ u.c.. Soluția x=(0, 9, 0, 20, 0, 0, 0, 21, 0, 0, 9, 0) este asociată cromozomului {{1→3,1→4,4→5,1→6},{2→3,2→4,2→5,2→6},{1,2},{1,2}} cu cele mai bune caracteristici care este soluția problemei după executarea a 3 iterații. STOP.

Anexa 9

Aplicarea algoritmului AME1

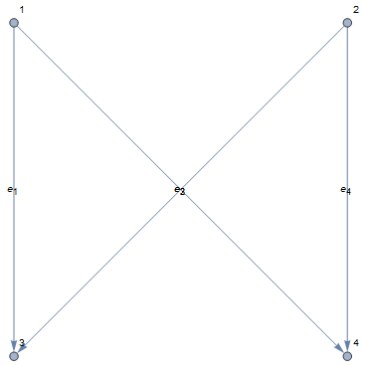


Fig. 9.1. Rețea cu două surse și două destinații

Sursa: elaborat de autor

Se consideră problema de transport descrisă de un graf bipartit dat în Figura 9.1, pentru care din sursele A_1 și A_2, vârfurile $\{1,2\}$ ale grafului, care conțin respectiv $\alpha_1 = 15$ și $\alpha_2 = 7$ cantități de produs de tipul P_1 și P_2 respectiv câte $\gamma_1 = 6$ și $\gamma_2 = 16$ unități fiecare. Produsele sunt transportate cu mijloacele de transport T_1 și T_2 de capacitatea respectivă $\delta_1 = 8$ și $\delta_2 = 14$ în destinațiile B_1 și B_2, vârfurile $\{3,4\}$ a grafului, care au necesitățile respective $\beta_1 = 12$ și $\beta_2 = 10$.

Costul de transport este descris de funcția $\varphi_1(x) = \begin{cases} 2x, & 0 \le x \le 4 \\ 8, & x > 4 \end{cases}$ pentru mulțimea de legături $\{(1111), (1122), (1211), (1212), (1222), (2121), (2211), (2212)\}$ iar funcția $\varphi_2(x) = \begin{cases} 3x, & 0 \le x \le 5 \\ 15, & x > 5 \end{cases}$ pentru mulțimea $\{(1112), (1121), (1221), (2111), (2112), (2122), (2221)\}$.

Pasul 1. Se determină o soluție admisibilă a problemei neliniare care satisface sistemul de restricții a problemei și se obține x^0_{ijkl} =(3, 0, 2, 7, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7).

Iterația 1.

Pasul 2. Se calculează coeficienții C_{ijkl}=(2, 3, 3, 8/7, 2, 2, 3, 2, 3, 3, 2, 3, 2, 2, 3, 8/7) și se determină valoarea funcției neliniare $F(x^0) = 34$.

Pasul 3. Se construiește funcția liniară $Z(x_{ijkl})$ și se soluționează problema liniară de transport $Z(x_{ijkl}) = 2x_{1111} + 3x_{1112} + 3x_{1121} + \frac{8}{7}x_{1122} + 2x_{1211} + 2x_{1212} + 3x_{1221} + 2x_{1222} + 3x_{2111} + 3x_{2112} + 2x_{2121} + 3x_{2122} + 2x_{2211} + 2x_{2212} + 3x_{2221} + \frac{8}{7}x_{2222}$ cu restricțiile problemei neliniare.

Se obține soluția optimă x^1=(1, 0, 0, 9, 5, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 5) iar $Z(x^1) = 32$.

Pasul 4. Deoarece $Z(x^1) < F(x^0)$ se substituie x^0 cu x^1 și se trece la *Pasul 2*.

Iterația 2.

Pasul 2. Se calculează coeficienții $C_{ijkl} = (2, 3, 3, \frac{8}{9}, \frac{8}{5}, 2, 3, 2, 3, 3, 2, 3, 2, 2, 3, \frac{8}{5})$ și se determină valoarea funcției neliniare $F(x^0) = 30$.

Pasul 3. Se construiește funcția liniară $Z(x_{ijkl})$ și se soluționează problema liniară de transport $Z(x_{ijkl}) = 2x_{1111} + 3x_{1112} + 3x_{1121} + \frac{8}{9}x_{1122} + \frac{8}{5}x_{1211} + 2x_{1212} + 3x_{1221} + 2x_{1222} + 3x_{2111} + 3x_{2112} + 2x_{2121} + 3x_{2122} + 2x_{2211} + 2x_{2212} + 3x_{2221} + \frac{8}{5}x_{2222}$ cu restricțiile problemei neliniare.

Se obține soluția optimă x^1=(0, 0, 0, 10, 5, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 4) iar $Z(x^1) = \frac{1318}{45}$.

Pasul 4. Deoarece $Z(x^1) < F(x^0)$ se substituie x^0 cu x^1 și se trece la *Pasul 2*.

Iterația 3.

Pasul 2. Se calculează coeficienții C_{ijkl}= (2, 3, 3, 4/5, 8/5, 2, 3, 2, 3, 3, 2, 3, 2, 2, 3, 2) și se determină valoarea funcției neliniare $F(x^0) = 30$.

Pasul 3. Se construiește funcția liniară $Z(x_{ijkl})$ și se soluționează problema liniară de transport $Z(x_{ijkl}) = 2x_{1111} + 3x_{1112} + 3x_{1121} + \frac{4}{5}x_{1122} + \frac{8}{5}x_{1211} + 2x_{1212} + 3x_{1221} + 2x_{1222} + 3x_{2111} + 3x_{2112} + 2x_{2121} + 3x_{2122} + 2x_{2211} + 2x_{2212} + 3x_{2221} + 2x_{2222}$ cu restricțiile problemei neliniare.

Se obține soluția optimă $x^1 = (0, 0, 0, 10, 5, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 4)$ iar $Z(x^1) = 30$.

Pasul 4. Deoarece $Z(x^1) = F(x^0)$ și $x^1 = x^0$ atunci $x^* = x^1$=(0, 0, 0, 10, 5, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 4) este soluție optimă.

Deci, din sursa 1 în destinația 1 este transportat un flux de produs de 10 u. c. de tipul 2 cu transportul 2, din sursa 1 în destinația 2 este transportat un flux de produs de 5 u. c. de tipul 1 cu transportul 1, din sursa 2 în destinația 1 este transportat un flux de produs de 2 u. c. de tipul 2 cu transportul 1, din sursa 2 în destinația 2 este transportat un flux de produs de 1 u. c. de tipul 1 cu transportul 1, din sursa 2 în destinația 2 este transportat un flux de produs de 4 u. c. de tipul 2 cu transportul 2 cu un cost total de transport de 30 u. c.

Se observă, că deși algoritmul pornește de la o soluție pentru care avem $F(x^0) = 34$ u.c. după 3 iterații se obține o micșorare a costului cheltuielilor de transport până la $F(x^*) = 30$ u.c..

Anexa 10

Aplicarea algoritmului AMG1

Se consideră problema de transport descrisă de un graf bipartit dat în Figura 9.1, pentru care din sursele A_1 și A_2, vârfurile $\{1,2\}$ ale grafului, care conțin respectiv $\alpha_1 = 31$ și $\alpha_2 = 69$ cantități de produs de tipul P_1, P_2 și P_3 respectiv câte $\gamma_1 = 28$, $\gamma_2 = 23$ și $\gamma_3 = 49$ unități fiecare. Produsele sunt transportate cu ajutorul mijloacelor de transport T_1, T_2 și T_3 de capacitatea respectivă $\delta_1 = 76$, $\delta_2 = 7$ și $\delta_3 = 17$ în destinațiile B_1 și B_2, vârfurile $\{3,4\}$ a grafului, care au necesitățile respective $\beta_1 = 38$ și $\beta_2 = 62$.

Costul de transport este descris de funcția $\varphi_1(x) = \begin{cases} x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$ pentru mulțimea de legături $\{(1111), (1112), (1121), (1122), (1123), (1133), (1212), (1222), (1223), (1232), (2112), (2113), (2123), (2132), (2133), (2211), (2212), (2213), (2222), (2231), (2232)\}$ iar funcția $\varphi_2(x) = \begin{cases} 2x, & 0 \le x \le 2 \\ 4, & x > 2 \end{cases}$ pentru mulțimea $\{(1113), (1131), (1132), (1211), (1213), (1221), (1231), (1233), (2111), (2121), (2122), (2131), (2221), (2223), (2233)\}$. Asupra cromozomilor-urmași este aplicată mutația cu o probabilitate 0,01.

Pasul 1. Se generează populația inițială din 16 cromozomi și se calculează valoarea funcției obiectiv asociată fiecărui cromozom din populație:

12,{1,4,2,3},{1,2,3},{1,3,2}},
12,{4,1,2,3},{3,2,1},{3,1,2}},
13,{1,3,4,2},{2,1,3},{2,1,3}},
13,{2,3,1,4},{2,3,1},{3,2,1}},
16,{1,2,3,4},{3,1,2},{1,2,3}},
16,{2,1,4,3},{3,1,2},{1,3,2}},
16,{3,1,4,2},{2,1,3},{3,2,1}},
17,{1,2,3,4},{1,3,2},{1,3,2}},
19,{1,2,4,3},{3,1,2},{3,2,1}},
19,{1,3,2,4},{1,3,2},{2,1,3}},
19,{2,4,1,3},{3,2,1},{2,1,3}},
19,{3,1,2,4},{1,2,3},{3,1,2}},
19,{3,1,4,2},{2,3,1},{3,1,2}},
19,{3,4,1,2},{3,2,1},{2,3,1}},
20,{3,2,1,4},{1,3,2},{1,3,2}},
22,{2,1,3,4},{1,3,2},{3,2,1}}.

Valoarea funcției obiectiv totală pentru populația inițială este $F_T(x) = 271$ u. c..

Iterația 1, se execută succesiv pașii 2 – 6.

Cromozomii-părinți care participă la încrucișare:	Cromozomii-urmași obținuți după încrucișare:
12,{1,4,2,3},{1,2,3},{1,3,2}},	{1,4,2,3},{1,2,3},{1,3,2}},
12,{4,1,2,3},{3,2,1},{3,1,2}},	{4,1,2,3},{3,2,1},{1,3,2}},
13,{1,3,4,2},{2,1,3},{2,1,3}},	{1,3,4,2},{2,1,3},{2,3,1}},
13,{2,3,1,4},{2,3,1},{3,2,1}},	{1,3,4,2},{2,3,1},{2,1,3}},
16,{1,2,3,4},{3,1,2},{1,2,3}},	{1,2,4,3},{3,2,1},{1,3,2}},
16,{2,1,4,3},{3,1,2},{1,3,2}},	{1,2,4,3},{3,1,2},{1,2,3}},
16,{3,1,4,2},{2,1,3},{3,2,1}},	{3,1,2,4},{2,3,1},{1,3,2}},
17,{1,2,3,4},{1,3,2},{1,3,2}},	{1,2,3,4},{1,3,2},{3,2,1}}}.

Populația obținută și valoarea funcției obiectiv pentru fiecare cromozom este:

11,{{1,3,4,2},{2,1,3},{2,3,1}},
12,{{1,2,4,3},{3,2,1},{1,3,2}},
12,{{1,4,2,3},{1,2,3},{1,3,2}},
12,{{1,4,2,3},{1,2,3},{1,3,2}},
12,{{4,1,2,3},{3,2,1},{1,3,2}},
12,{{4,1,2,3},{3,2,1},{3,1,2}},
13,{{1,2,4,3},{3,1,2},{1,2,3}},
13,{{1,3,4,2},{2,1,3},{2,1,3}},
13,{{1,3,4,2},{2,3,1},{2,1,3}},
13,{{2,3,1,4},{2,3,1},{3,2,1}},
16,{{1,2,3,4},{3,1,2},{1,2,3}},
16,{{2,1,4,3},{3,1,2},{1,3,2}},
16,{{3,1,2,4},{2,3,1},{1,3,2}},
16,{{3,1,4,2},{2,1,3},{3,2,1}},
17,{{1,2,3,4},{1,3,2},{1,3,2}},
19,{{1,2,3,4},{1,3,2},{3,2,1}}

Pentru cromozomul cu cele mai bune caracteristici {{1,3,4,2},{2,1,3},{2,3,1}} îi este asociată soluția x = (7, 0, 1, 0, 7, 16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 13, 0, 0, 0, 0, 0, 49, 0, 0). Valoarea funcției obiectiv este $F(x) = 11$ u. c. și valoarea funcției obiectiv totală pentru populația obținută este $F_T(x) = 223$ u. c.

Iterația 2, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 8 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Soluția x = (7, 1, 0, 0, 6, 17, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 13, 0, 0, 0, 0, 0, 49, 0, 0) este asociată cromozomului {{1,3,4,2},{2,1,3},{3,1,2}} cu cele mai bune caracteristici. Valoarea funcției obiectiv este $F(x) = 10$ u. c. și valoarea funcției obiectiv totală pentru populația obținută este $F_T(x) = 204$ u. c..

Iterația 3, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 8 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Soluția x= (0, 0, 0, 23, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 4, 0, 17, 0, 0, 0, 41, 0, 0) este asociată cromozomului {{1,4,2,3},{2,3,1},{1,3,2}} cu cele mai bune caracteristici. Valoarea funcției obiectiv este $F(x) = 9$ u. c. și valoarea funcției obiectiv totală pentru populația obținută este $F_T(x) = 188$ u. c..

Dacă sunt executate 3 iterații atunci soluția obținută este soluția problemei.

Anexa 11

Aplicarea algoritmului AE2

Se consideră problema de transport pe rețea pentru care se cunoaște că sursele A_1, A_2 și A_3 conțin respectiv $\alpha_1 = 22$, $\alpha_2 = 12$ și $\alpha_3 = 16$ cantități de produs de tipul P_1 și P_2 respectiv câte $\gamma_1 = 7$ și $\gamma_2 = 43$ unități fiecare. Produsele sunt transportate cu mijloacele de transport T_1, T_2 și T_3 de capacitatea respectivă $\delta_1 = 29$, $\delta_2 = 19$ și $\delta_3 = 2$ în destinațiile B_1 și B_2 care au necesitățile respective $\beta_1 = 29$ și $\beta_2 = 31$.

Rețeaua de transport din Figura 11.1 este descrisă de mulțimea de arce: {1→4, 1→5, 1→6, 1→7, 1→8, 2→4, 2→5, 2→6, 2→7, 2→8, 3→4, 3→5, 3→6, 3→7, 3→8, 4→5, 4→6, 4→7, 4→8, 5→6, 5→7, 5→8, 6→7, 6→8}, unde {1, 2, 3} – mulțimea surselor, {7, 8} – mulțimea destinațiilor.

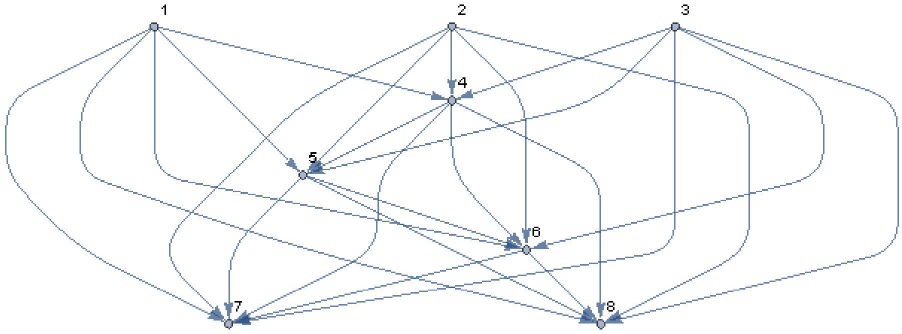


Fig. 11.1 Rețea de transport cu 3 surse și 2 destinații Sursa: elaborat de autor

Costul de transport este descris de funcțiile: $\varphi_1(x) = \begin{cases} x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$ și $\varphi_2(x) = \begin{cases} 2x, & 0 \le x \le 2 \\ 4, & x > 2 \end{cases}$.

Pasul 1. Se determină o soluție admisibilă a problemei neliniare. Se obține x_{ekl}^0=(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 14, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).

Iterația 1.

Pasul 2. Se calculează coeficienții C_{ekl} = (2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1, 4/7, 1, 1, 4/15,2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 4/7,2, 2, 1, 2, 2, 2, 1/5,1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2/7,2, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 2) și $F(x^0) = 21$.

Pasul 3. Se construiește funcția liniară $Z(x_{ekl})$ și se soluționează problema liniară de transport cu aceleași restricții. Se obține soluția optimă x^1 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 5, 0, 0, 17, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) iar $Z(x^1) = 18.39$.

Pasul 4. Deoarece $Z(x^1) < F(x^0)$ se substituie x^0 cu x^1 și se trece la *Pasul 2*.

Iterația 2.

Pasul 2. Se calculează coeficienții $C_{ekl} =$ (2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1, 4/5, 1, 1, 4/17,2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 4/7,2, 2, 1, 2, 2, 2, 1/5, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 2, 2, 2, 1/2,2, 2/7,2, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 2) și $F(x^0) = 18$.

Pasul 3. Se construiește funcția liniară $Z(x_{ekl})$ și se soluționează problema liniară de transport cu aceleași restricții. Se obține soluția optimă x^1 =(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0, 0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0, 5, 0, 0, 0, 0, 0, 0, 0, 2, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) iar $Z(x^1) = 16.89$.

Pasul 4. Deoarece $Z(x^1) < F(x^0)$ se substituie x^0 cu x^1 și se trece la *Pasul 2*.

Iterația 3.

Pasul 2. Se calculează coeficienții C_{ekl}= 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1, 2/11,2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1/10,1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1/5,1, 1, 2, 1, 2, 2, 2, 1/2,2, 4/9,2, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 2) și $F(x^0) = 15$.

Pasul 3. Se construiește funcția liniară $Z(x_{ekl})$ și se soluționează problema liniară de transport cu aceleași restricții. Se obține soluția optimă x^1=(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) iar $Z(x^1) = 12.4$.

Pasul 4. Deoarece $Z(x^1) < F(x^0)$ se substituie x^0 cu x^1 și trecem la *Pasul 2*.

Iterația 4.

Pasul 2. Se calculează coeficienții C_{ekl}=(2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1, 2/11, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1/10, 1/2, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1/7, 1, 1, 2, 1, 2, 2, 2, 1, 2, 4/9, 2, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 2) și $F(x^0) = 11$.

Se construiește funcția liniară $Z(x_{ekl})$ și se soluționează problema liniară de transport cu aceleași restricții. Se obține soluția optimă x^1=(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) iar $Z(x^1) = 11$.

Pasul 4. Deoarece $Z(x^1) = F(x^0)$ și $x^1 = x^0$ atunci $x^* = x^1 =$ (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) este soluție optimă.

Deci, din sursa 1 în destinația 1 este transportat un flux de produs de 22 u. c. de tipul 2 cu transportul 1, din sursa 2 în destinația 2 este transportat un flux de produs de 10 u. c. de tipul 2 cu transportul 2, din sursa 2 în destinația 2 este transportat un flux de produs de 2 u. c. de tipul 2 cu transportul 3, din sursa 3 în destinația 1 este transportat un flux de produs de 7 u. c. de tipul 1 cu transportul 1, din sursa 3 în destinația 2 este transportat un flux de produs de 9 u. c. de tipul 2 cu transportul 2 cu un cost total de transport de 11 u. c.

Anexa 12

Aplicarea algoritmului AMG2

Se consideră problema de transport prezentată în Figura12.1 descrisă de sursele A_1 și A_2 care conțin respectiv $\alpha_1 = 34$ și $\alpha_2 = 16$ cantități de produs de tipul P_1 și P_2 respectiv câte $\gamma_1 = 37, \gamma_2 = 13$ unități fiecare. Produsele sunt transportate cu mijloacele de transport T_1 și T_2 de capacitatea respectivă $\delta_1 = 30$ și $\delta_2 = 20$ în destinațiile B_1 și B_2 care au necesitățile respective $\beta_1 = 46$ și $\beta_2 = 4$.

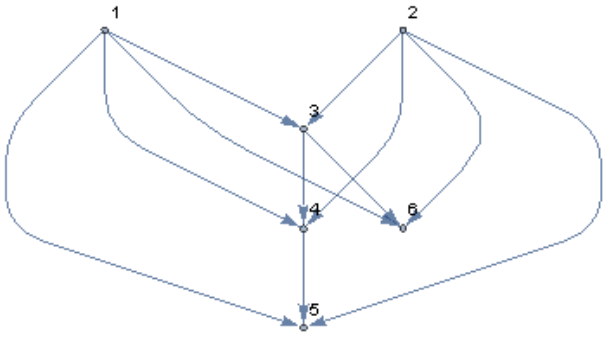


Fig. 12.1 Rețea de transport două surse și două destinații Sursa: elaborat de autor

Costul de transport este descris de funcția $\varphi_1(x) = \begin{cases} x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$ pentru mulțimea de legături $\{(111), (122), (221), (311), (312), (321), (422), (511), (512), (622), (712), (721), (722), (811), (812), (821), (822), (912), (921), (10\ 11), (11\ 11), (11\ 21)\}$ și funcția $\varphi_2(x) = \begin{cases} 2x, & 0 \le x \le 2 \\ 4, & x > 2 \end{cases}$ pentru mulțimea de legături $\{(112), (121), (211), (212), (222), (322), (411), (412), (421), (521), (522), (611), (612), (621), (711), (911), (922), (10\ 12), (10\ 21), (10\ 22), (11\ 12), (11\ 22)\}$. Asupra cromozomilor-urmași este aplicată mutația cu o probabilitate 0,01.

Pasul 1 Se generează populația inițială din 24 cromozomi:

{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,3→6}},{2,1},{1,2},{1,2},{1,2}},
{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{1,2},{1,2}},
{{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{1,2},{1,2},{2,1}},
{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,3→6}},{1,2},{1,2},{1,2},{1,2}},
{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{1,2},{1,2}},
{{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,3→6}},{2,1},{2,1},{1,2},{1,2}},
{{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{1,2},{1,2}},
{{1→3,3→4,4→5,1→6},{2→3,2→4,4→5,3→6}},{2,1},{2,1},{2,1},{1,2}},
{{1→3,3→4,4→5,3→6},{2→3,3→4,4→5,2→6}},{1,2},{1,2},{1,2},{1,2}},
{{1→3,1→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{2,1},{1,2},{1,2},{2,1}},
{{1→3,1→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{2,1},{2,1}},
{{1→3,3→4,4→5,3→6},{2→3,2→4,2→5,2→6}},{2,1},{2,1},{2,1},{1,2}},
{{1→3,3→4,4→5,3→6},{2→3,2→4,4→5,2→6}},{1,2},{2,1},{2,1},{1,2}},
{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{2,1},{2,1}},
{{1→3,1→4,4→5,1→6},{2→3,3→4,2→5,3→6}},{1,2},{2,1},{1,2},{2,1}},
{{1→3,1→4,4→5,3→6},{2→3,2→4,2→5,3→6}},{1,2},{1,2},{2,1},{2,1}},
{{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,3→6}},{2,1},{1,2},{2,1},{1,2}},

{{1→3,1→4,4→5,3→6},{2→3,3→4,4→5,3→6}},{2,1},{2,1},{1,2},{1,2}},
{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,2→6}},{1,2},{2,1},{2,1},{1,2}},
{{1→3,3→4,1→5,1→6},{2→3,2→4,2→5,3→6}},{1,2},{2,1},{2,1},{1,2}},
{{1→3,3→4,4→5,3→6},{2→3,3→4,2→5,3→6}},{2,1},{2,1},{2,1},{2,1}},
{{1→3,3→4,4→5,1→6},{2→3,2→4,4→5,2→6}},{2,1},{1,2},{2,1},{1,2}},
{{1→3,3→4,4→5,3→6},{2→3,2→4,4→5,2→6}},{1,2},{1,2},{1,2},{2,1}},
{{1→3,1→4,4→5,3→6},{2→3,2→4,4→5,2→6}},{2,1},{2,1},{1,2},{2,1}}}

Iterația 1, se execută succesiv pașii 2 – 6. Sun selectați în calitate de cromozomi-părinții 12 cromozomi pentru care valorile funcțiilor obiectiv în soluțiile asociate sunt minime:

{8,{{1→3,3→4,1→5,1→6},{2→3,2→4,2→5,3→6}},{1,2},{2,1},{2,1},{1,2}},
{8,{{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{1,2},{1,2},{2,1}},
{13,{{1→3,1→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{2,1},{1,2},{1,2},{2,1}},
{13,{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{1,2},{1,2}},
{16,{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,3→6}},{2,1},{1,2},{1,2},{1,2}},
{19,{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,2→6}},{1,2},{2,1},{2,1},{1,2}},
{20,{{1→3,1→4,4→5,3→6},{2→3,2→4,4→5,2→6}},{2,1},{2,1},{1,2},{2,1}},
{21,{{1→3,1→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{2,1},{2,1}},
{21,{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{2,1},{2,1}},
{21,{{1→3,3→4,4→5,3→6},{2→3,2→4,2→5,2→6}},{2,1},{2,1},{2,1},{1,2}},
{22,{{1→3,1→4,4→5,1→6},{2→3,3→4,2→5,3→6}},{1,2},{2,1},{1,2},{2,1}},
{24,{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{1,2},{1,2}},

Cromozomii-urmașii obținuți după încrucișare sunt:

{{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{1,2},{1,2},{2,1}},
{{1→3,3→4,1→5,1→6},{2→3,2→4,2→5,3→6}},{1,2},{2,1},{2,1},{1,2}},
{{1→3,1→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{1,2},{1,2},{1,2}},
{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{1,2},{1,2},{2,1},{1,2}},
{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,2→6}},{2,1},{2,1},{1,2},{1,2}},
{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,3→6}},{2,1},{2,1},{2,1},{2,1}},
{{1→3,1→4,4→5,3→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{1,2},{2,1}},
{{1→3,1→4,1→5,3→6},{2→3,2→4,4→5,2→6}},{2,1},{1,2},{1,2},{1,2}},
{{1→3,3→4,1→5,3→6},{2→3,2→4,2→5,2→6}},{1,2},{2,1},{1,2},{2,1}},
{{1→3,3→4,4→5,3→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{2,1},{2,1}},
{{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{1,2},{2,1}},
{{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,3→6}},{1,2},{1,2},{1,2},{1,2}}}

Populația obținută și valoarea funcției obiectiv pentru fiecare cromozom:

{{8,{{1→3,3→4,1→5,1→6},{2→3,2→4,2→5,3→6}},{1,2},{2,1},{2,1},{1,2}},
{8,{{1→3,3→4,1→5,1→6},{2→3,2→4,2→5,3→6}},{1,2},{2,1},{2,1},{1,2}},
{8,{{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{1,2},{1,2},{2,1}},
{8,{{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{1,2},{1,2},{2,1}},
{10,{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{1,2},{1,2},{2,1},{1,2}},
{12,{{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,3→6}},{1,2},{1,2},{1,2},{1,2}},
{12,{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,2→6}},{2,1},{2,1},{1,2},{1,2}},
{13,{{1→3,1→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{2,1},{1,2},{1,2},{2,1}},
{13,{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{1,2},{1,2}},
{15,{{1→3,3→4,1→5,3→6},{2→3,2→4,2→5,2→6}},{1,2},{2,1},{1,2},{2,1}},
{16,{{1→3,1→4,1→5,3→6},{2→3,2→4,4→5,2→6}},{2,1},{1,2},{1,2},{1,2}},
{16,{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,3→6}},{2,1},{1,2},{1,2},{1,2}},
{18,{{1→3,1→4,4→5,1→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{1,2},{2,1}},
{19,{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,2→6}},{1,2},{2,1},{2,1},{1,2}},
{20,{{1→3,1→4,4→5,3→6},{2→3,2→4,4→5,2→6}},{2,1},{2,1},{1,2},{2,1}},
{21,{{1→3,1→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{1,2},{1,2},{1,2}},
{21,{{1→3,1→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{2,1},{2,1}},
{21,{{1→3,3→4,1→5,3→6},{2→3,3→4,4→5,2→6}},{2,1},{2,1},{2,1},{2,1}},
{21,{{1→3,3→4,4→5,3→6},{2→3,2→4,2→5,2→6}},{2,1},{2,1},{2,1},{1,2}},
{22,{{1→3,1→4,4→5,1→6},{2→3,3→4,2→5,3→6}},{1,2},{2,1},{1,2},{2,1}},

{23,{{1→3,3→4,1→5,3→6},{2→3,2→4,4→5,3→6}},{2,1},{2,1},{2,1},{2,1}},
{24,{{1→3,3→4,1→5,1→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{1,2},{1,2}},
{29,{{1→3,3→4,4→5,3→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{2,1},{2,1}},
{32,{{1→3,1→4,4→5,3→6},{2→3,3→4,4→5,2→6}},{1,2},{2,1},{1,2},{2,1}}}

Cromozomului cu cele mai bune caracteristici {{1→3,3→4,1→5,1→6},{2→3,2→4, 2→5, 3→6}},{1,2},{2,1},{2,1},{1,2}} îi este asociată soluția x=(0, 0, 0, 0, 0, 0, 0, 0, 17, 4, 9, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). Valoarea funcției obiectiv este $F(x) = 8$ u. c. și valoarea funcției obiectiv totală pentru populația obținută este $F_T(x) = 410$ u. c.

Iterația 2, se execută succesiv pașii 2 – 6. Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 12 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Soluția x= (0, 0, 0, 0, 0, 0, 0, 0, 17, 4, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) este asociată cromozomului {1→3,3→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{2,1},{2,1},{1,2},{2,1}} cu cele mai bune caracteristici. Valoarea funcției obiectiv este $F(x) = 5$ u. c. și valoarea funcției obiectiv totală pentru populația obținută este $F_T(x) = 309$ u. c..

Iterația 3, se execută succesiv pașii 2 – 6. . Cromozomii populației sunt sortați în ordine crescătoare a valorii funcției obiectiv în soluția asociată cromozomilor respectivi. Primii 12 cromozomi sunt transferați în următoarea populație și participă la încrucișare. Soluția x= (0, 0, 0, 0, 0, 0, 0, 0, 17, 4, 13, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) asociată cromozomului {{1→3,3→4,1→5,1→6},{2→3,3→4,2→5,2→6}},{1,2},{1,2},{2,1},{1,2}}. Valoarea funcției obiectiv este $F(x) = 5$ u. c. și valoarea funcției obiectiv totală pentru populația obținută este $F_T(x) = 244$ u. c..

Dacă sunt executate 3 iterații atunci soluția obținută este soluția problemei.

DECLARAȚIA PRIVIND ASUMAREA RĂSPUNDERII

Subsemnata, declar pe răspundere personală că materialele prezentate în teza de doctorat sunt rezultatul propriilor cercetări și realizări științifice. Conștientizez că, în caz contrar, urmează să suport consecințele în conformitate cu legislația în vigoare.

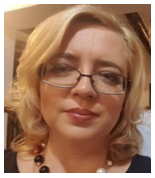
Numele de familie, prenumele Rașe Svetlana

Semnătura

Data 9.11.2021

CURRICULUM VITAE

INFORMAŢII PERSONALE



Tatiana Pașa

4/1, N. Dimo, MD 2068, Chișinău, RM

(+373) 79406452

pasa.tatiana@yahoo.c

EXPERIENŢA PROFESIONALĂ

01/09/1999 - prezent	Lector universitar, Universitatea de Stat din Moldova, Chișinău, RM
16/10/2020 - prezent	Președinta Comitetului Sindical al Facultății de Matematică și Informatică

EDUCAŢIE ŞI FORMARE

01/01/2018–31/12/2018	Studii doctorale, Cibernetică Matematică și Cercetări Operaționale USM, Facultatea de Matematică și Informatică, Cișinău, RM
01/09/2012–18/06/2014	Master în Științe Economice, Administrarea Afacerilor Universitatea de Stat din Moldova, Chișinău, RM
01/09/1995–01/07/2000	Licențiat în Matematică, specialitatea Matematică și Informatică Universitatea de Stat din Moldova, Chișinău, RM

COMPETENŢE PERSONALE

Limbi română**,** rusa, engleza

Afiliere

- Wolfram Center Moldova
- LMPI - N°573901-EPP-1-2016-1-IT-EPPKA2-CBHE-JP, *"Licence, Master professionnels pour le développement, l'administration, la gestion, la protection des systèmes et réseaux informatiques dans les entreprises en Moldavie, au Kazakhstan, au Vietnam"*
- LCȘ "Inteligență Artificială și Realitate Virtuală & Augmentată" al Departamentului Informatică, USM

Cursuri on-line

1. Pașa T., *TIC*, 2014, http://moodle.usm.md/moodle/ course/view.php?id=890
2. Pașa T., *Securitatea Informației Întreprinderii*, 2018, http://moodle.usm.md/moodle/course/view.php?id=1528
3. Pașa T., *Securitatea Bazelor de Date*, 2018, http://moodle.usm.md/moodle/course/view.php?id=1525
4. Pașa T., *HTML CSS JS*, 2019, http://moodle.usm.md/moodle/course/view.php?id=1699
5. Pașa T., *POO*, 2020, https://moodle.usm.md/course/view.php?id=2093

Conferințe

1. A 20-a Conferință a SPSR, AFA "Henri Coandă", Departamentul de Științe Fundamentale și Management, 28-29 aprilie 2017, Brașov, România.
2. The Fourth Conference of Mathematical Society of the Republic of Moldova dedicated to the centenary of Vladimir Andrunachevici (1917 - 1997), 28 iunie – 2 iulie 2017, Institute of Mathematics and Computer Science, Academy of Scieces of Moldova, Chișinău.

3. The 50th anniversary of Computers, Informatics and Microelectronics Faculty & Electronics and Telecommunications Faculty, October 19-21, 2017, Chisinau, RM.
4. A 21-a Conferință a Societății de Probabilități și Statistică din România, Academia de Studii Economice din București, 13-14 aprilie 2018, ASE, București, România.
5. International Conference on Mathematics, Informatics and Information technologies dedicated to the illustrious scientist Valentin Belousov, 19-21 April, 2018, Bălți, RM.
6. CAIM 2018, The 26th Conference on Applied and Industrial Mathematics, 20th – 23th September, 2018, Technical University of Moldova, Chișinău, Moldova.
7. CAIM 2019, The 27th Conference on Applied and Industrial Mathematics, 19th – 20th September, 2019, "Valahia" University, Târgoviște, Romania.
8. IMCS - 55, The Fifth Conference of Mathematical Society of the Republic of Moldova, September 28 - October 1, 2019, Chișinău, Moldova.
9. International Conference on Applied and Pure Mathematics, 6th edition, ICAPM 2019, October 31 – November 3, Iași, România.
10. International Conference on Applied Mathematics and Numerical Methods, Third Edition, Craiova, October 29-31, 2020.
11. International Symposium ″Actual Problems of Mathematics and Informatics″ dedicated to the 90th Birthday of Professor Ion Valuță, November 27-28, 2020, TUM, Chișinău, RM.

Publicaţii

1. **PAŞA, T.**, UNGUREANU, V., Wolfram Mathematica as an enviroment for solving concave network transportation problem. In: *The Fourth Conference of Mathematical Society of the Republic of Moldova dedicated to the centenary of Vladimir Andrunachevici (1917 - 1997), 28 iunie - 2 iulie 2017*, p. 429 - 432. Chișinău: IMCS, AŞM. ISBN 978 - 9975 - 71 - 915 - 5.
2. **PAŞA, T.**, UNGUREANU, V., Non-Linear Concave Transportation Problem Solving and Implementation using Wolfram Language., In: *ITSN, 2017,* p. 30-39. Chișinău. ISBN 978-9975-3168-5-9.
3. **PAŞA, T.**, UNGUREANU, V., Applying sequential and parallel programming to solve a non-linear transport problem. In: *ICMCS, October 19 - 21, 2017*, p. 247-251. Cișinău, Republic of Moldova. ISBN 978-9975-4264-8-0.
4. **PAŞA, T.**, Multi-index transport problem with non-linear cost functions. In: *Romai J., 2018, vol. 14,* no. 2, p. 129-137. Disponibil: https://rj.romai.ro/arhiva/2018/2/Pasa.pdf.
5. **PAŞA, T.,** UNGUREANU, V., Solving the non-linear 4-index transportation problem. In: *The Fifth Conference of Mathematical Society of the Republic of Moldova, 2019,* p. 221-224. Chișinău: Vladimir Andrunachievici Institute of Mathematics and Computer Science.
6. **PAŞA, T.**, Solving non-linear multi-index transportation problems. In: *Romai J., 2019, vol. 15,* no. 2, p. 91-99. Disponibil: https://rj.romai.ro/arhiva/2019/2/Pasa.pdf.
7. **PAŞA, T.**, Solving transportation problems with concave cost functions using genetic algorithms. In: *CSJM, 2020, vol. 28 no. 2* (83)*,* p.140-151. ISSN 1561-4042.
8. **PAŞA, T.,** Genetic algorithm for solving transportation problems on networks with one source and multiple sinks. In: *ITM Web Conf., ICAMNM 2020, Section: Applied Mathematics and Numerical Methods, nr. 34*, 11 pages. https://doi.org/10.1051/itmconf/20203402006, Online ISSN 2271-2097.