Calcul inteligent şi rezolvarea problemelor

P.C. Pop[1]

[1]Departmentul de Matematică şi Informatică

Universitatea Tehnică Cluj-Napoca, Centrul Universitar Nord din Baia Mare

Outline

Rezolvarea problemelor

- Rezolvarea unei probleme înseamnă stabilirea unei asocieri între datele de intrare şi răspunsul corect.
- De multe ori, o problemă nu are o singură soluţie.
- În majoritatea situaţiilor este importantă găsirea soluţiei optime care rezolvă problema cu maximă eficienţă.
- Pentru rezolvarea unei probleme generale trebuie respectaţi următorii paşi:
  - definirea cât mai precisă a problemei: specificaţii ale stării iniţiale şi finale (condiţii şi soluţii fezabile);
  - analiza problemei;
  - alegerea celei mai potrivite tehnici de rezolvare şi aplicarea ei la problema curentă.

Rezolvarea problemelor

- Rezolvarea unei probleme înseamnă stabilirea unei asocieri între datele de intrare şi răspunsul corect.
- De multe ori, o problemă nu are o singură soluţie.
- În majoritatea situaţiilor este importantă găsirea soluţiei optime care rezolvă problema cu maximă eficienţă.
- Pentru rezolvarea unei probleme generale trebuie respectaţi următorii paşi:
  - definirea cât mai precisă a problemei: specificaţii ale stării iniţiale şi finale (condiţii şi soluţii fezabile);
  - analiza problemei;
  - alegerea celei mai potrivite tehnici de rezolvare şi aplicarea ei la problema curentă.

Rezolvarea problemelor

- Rezolvarea unei probleme înseamnă stabilirea unei asocieri între datele de intrare şi răspunsul corect.
- De multe ori, o problemă nu are o singură soluţie.
- În majoritatea situaţiilor este importantă găsirea soluţiei optime care rezolvă problema cu maximă eficienţă.
- Pentru rezolvarea unei probleme generale trebuie respectaţi următorii paşi:
  - definirea cât mai precisă a problemei: specificaţii ale stării iniţiale şi finale (condiţii şi soluţii fezabile);
  - analiza problemei;
  - alegerea celei mai potrivite tehnici de rezolvare şi aplicarea ei la problema curentă.

Rezolvarea problemelor

- Rezolvarea unei probleme înseamnă stabilirea unei asocieri între datele de intrare şi răspunsul corect.
- De multe ori, o problemă nu are o singură soluţie.
- În majoritatea situaţiilor este importantă găsirea soluţiei optime care rezolvă problema cu maximă eficienţă.
- Pentru rezolvarea unei probleme generale trebuie respectaţi următorii paşi:
  - definirea cât mai precisă a problemei: specificaţii ale stării iniţiale şi finale (condiţii şi soluţii fezabile);
  - analiza problemei;
  - alegerea celei mai potrivite tehnici de rezolvare şi aplicarea ei la problema curentă.

Rezolvarea problemelor

- Rezolvarea unei probleme înseamnă stabilirea unei asocieri între datele de intrare şi răspunsul corect.
- De multe ori, o problemă nu are o singură soluţie.
- În majoritatea situaţiilor este importantă găsirea soluţiei optime care rezolvă problema cu maximă eficienţă.
- Pentru rezolvarea unei probleme generale trebuie respectaţi următorii paşi:
  - definirea cât mai precisă a problemei: specificaţii ale stării iniţiale şi finale (condiţii şi soluţii fezabile);
  - analiza problemei;
  - alegerea celei mai potrivite tehnici de rezolvare şi aplicarea ei la problema curentă.

Rezolvarea problemelor

- Rezolvarea unei probleme înseamnă stabilirea unei asocieri între datele de intrare şi răspunsul corect.
- De multe ori, o problemă nu are o singură soluţie.
- În majoritatea situaţiilor este importantă găsirea soluţiei optime care rezolvă problema cu maximă eficienţă.
- Pentru rezolvarea unei probleme generale trebuie respectaţi următorii paşi:
  - definirea cât mai precisă a problemei: specificaţii ale stării iniţiale şi finale (condiţii şi soluţii fezabile);
  - analiza problemei;
  - alegerea celei mai potrivite tehnici de rezolvare şi aplicarea ei la problema curentă.

Rezolvarea problemelor

- Din punctul de vedere al rezolvării automate problemele pot fi clasificate în două categorii:
  - Probleme bine definite: caracterizate prin faptul că li se poate asocia un model formal (de exemplu, un model matematic) pe baza căruia se poate dezvolta o metodă de rezolvare algoritmică.
  - Probleme rău definite: caracterizate prin faptul că nu pot fi descrise complet printr-un model formal, ci cel mult se cunosc răspunsuri pentru cazuri particulare ale problemei.
- A. Einstein spunea că: "If I had only one hour to save the world, I would spend fifty-five minutes defining the problem, and on;y five minutes finding the solution".
- Concluzie: Nu vă gândiţi numai la soluţii, gândiţi-vă la problemă: găsirea soluţiilor implică efort, în timp ce descrierea problemelor necesită inteligenţă.

Rezolvarea problemelor

- Din punctul de vedere al rezolvării automate problemele pot fi clasificate în două categorii:
  - Probleme bine definite: caracterizate prin faptul că li se poate asocia un model formal (de exemplu, un model matematic) pe baza căruia se poate dezvolta o metodă de rezolvare algoritmică.
  - Probleme rău definite: caracterizate prin faptul că nu pot fi descrise complet printr-un model formal, ci cel mult se cunosc răspunsuri pentru cazuri particulare ale problemei.
- A. Einstein spunea că: "If I had only one hour to save the world, I would spend fifty-five minutes defining the problem, and on;y five minutes finding the solution".
- Concluzie: Nu vă gândiţi numai la soluţii, gândiţi-vă la problemă: găsirea soluţiilor implică efort, în timp ce descrirea problemelor necesită inteligenţă.

Rezolvarea problemelor

- Din punctul de vedere al rezolvării automate problemele pot fi clasificate în două categorii:
  - Probleme bine definite: caracterizate prin faptul că li se poate asocia un model formal (de exemplu, un model matematic) pe baza căruia se poate dezvolta o metodă de rezolvare algoritmică.
  - Probleme rău definite: caracterizate prin faptul că nu pot fi descrise complet printr-un model formal, ci cel mult se cunosc răspunsuri pentru cazuri particulare ale problemei.
- A. Einstein spunea că: "If I had only one hour to save the world, I would spend fifty-five minutes defining the problem, and on;y five minutes finding the solution".
- Concluzie: Nu vă gândiţi numai la soluţii, gândiţi-vă la problemă: găsirea soluţiilor implică efort, în timp ce descrirea problemelor necesită inteligenţă.

Rezolvarea problemelor

- Din punctul de vedere al rezolvării automate problemele pot fi clasificate în două categorii:
  - Probleme bine definite: caracterizate prin faptul că li se poate asocia un model formal (de exemplu, un model matematic) pe baza căruia se poate dezvolta o metodă de rezolvare algoritmică.
  - Probleme rău definite: caracterizate prin faptul că nu pot fi descrise complet printr-un model formal, ci cel mult se cunosc răspunsuri pentru cazuri particulare ale problemei.
- A. Einstein spunea că: "If I had only one hour to save the world, I would spend fifty-five minutes defining the problem, and on;y five minutes finding the solution".
- Concluzie: Nu vă gândiţi numai la soluţii, gândiţi-vă la problemă: găsirea soluţiilor implică efort, în timp ce descrirea problemelor necesită inteligenţă.

Rezolvarea problemelor

- Din punctul de vedere al rezolvării automate problemele pot fi clasificate în două categorii:
  - Probleme bine definite: caracterizate prin faptul că li se poate asocia un model formal (de exemplu, un model matematic) pe baza căruia se poate dezvolta o metodă de rezolvare algoritmică.
  - Probleme rău definite: caracterizate prin faptul că nu pot fi descrise complet printr-un model formal, ci cel mult se cunosc răspunsuri pentru cazuri particulare ale problemei.
- A. Einstein spunea că: "If I had only one hour to save the world, I would spend fifty-five minutes defining the problem, and on;y five minutes finding the solution".
- Concluzie: Nu vă gândiţi numai la soluţii, gândiţi-vă la problemă: găsirea soluţiilor implică efort, în timp ce descrirea problemelor necesită inteligenţă.

Rezolvarea problemelor

- Problemele bine definite
  - Se caracterizează prin existenţa unui scop explicit şi a unor operatori cunoscuţi.
  - Cu alte cuvinte, avem o modalitate clară de rezolvare, toate elementele implicate în acest proces sunt specificate.
  - Sunt rezolvate prin metode standard au prin metode cunoscute pentru probleme similare sau analoage.
- Exemplu: rezolvarea unei ecuaţii algebrice de gradul întâi: $5x = 12$.
  - Ştim că pentru problema generală: $a \cdot x = b$, soluţia este: $x = \frac{b}{a}$, iar împărţirea este o operaţie cunoscută cu restricţia $a \neq 0$.
  - Deci în cazul nostru avem: $x = \frac{12}{5} = 2.4$.

Rezolvarea problemelor

- Problemele bine definite
  - Se caracterizează prin existenţa unui scop explicit şi a unor operatori cunoscuţi.
  - Cu alte cuvinte, avem o modalitate clară de rezolvare, toate elementele implicate în acest proces sunt specificate.
  - Sunt rezolvate prin metode standard au prin metode cunoscute pentru probleme similare sau analoage.
- Exemplu: rezolvarea unei ecuaţii algebrice de gradul întâi: $5x = 12$.
  - Ştim că pentru problema generală: $a \cdot x = b$, soluţia este: $x = \frac{b}{a}$, iar împărţirea este o operaţie cunoscută cu restricţia $a \neq 0$.
  - Deci în cazul nostru avem: $x = \frac{12}{5} = 2.4$.

Rezolvarea problemelor

- Problemele bine definite
  - Se caracterizează prin existenţa unui scop explicit şi a unor operatori cunoscuţi.
  - Cu alte cuvinte, avem o modalitate clară de rezolvare, toate elementele implicate în acest proces sunt specificate.
  - Sunt rezolvate prin metode standard au prin metode cunoscute pentru probleme similare sau analoage.
- Exemplu: rezolvarea unei ecuaţii algebrice de gradul întâi: $5x = 12$.
  - Ştim că pentru problema generală: $a \cdot x = b$, soluţia este: $x = \frac{b}{a}$, iar împărţirea este o operaţie cunoscută cu restricţia $a \neq 0$.
  - Deci în cazul nostru avem: $x = \frac{12}{5} = 2.4$.

Rezolvarea problemelor

- Problemele bine definite
  - Se caracterizează prin existenţa unui scop explicit şi a unor operatori cunoscuţi.
  - Cu alte cuvinte, avem o modalitate clară de rezolvare, toate elementele implicate în acest proces sunt specificate.
  - Sunt rezolvate prin metode standard au prin metode cunoscute pentru probleme similare sau analoage.
- Exemplu: rezolvarea unei ecuaţii algebrice de gradul întâi: $5x = 12$.
  - Ştim că pentru problema generală: $a \cdot x = b$, soluţia este: $x = \frac{b}{a}$, iar împărţirea este o operaţie cunoscută cu restricţia $a \neq 0$.
  - Deci în cazul nostru avem: $x = \frac{12}{5} = 2.4$.

Rezolvarea problemelor

- Problemele bine definite
  - Se caracterizează prin existenţa unui scop explicit şi a unor operatori cunoscuţi.
  - Cu alte cuvinte, avem o modalitate clară de rezolvare, toate elementele implicate în acest proces sunt specificate.
  - Sunt rezolvate prin metode standard au prin metode cunoscute pentru probleme similare sau analoage.
- Exemplu: rezolvarea unei ecuaţii algebrice de gradul întâi: $5x = 12$.
  - Ştim că pentru problema generală: $a \cdot x = b$, soluţia este: $x = \frac{b}{a}$, iar împărţirea este o operaţie cunoscută cu restricţia $a \neq 0$.
  - Deci în cazul nostru avem: $x = \frac{12}{5} = 2.4$.

Rezolvarea problemelor

- Problemele bine definite
  - Se caracterizează prin existenţa unui scop explicit şi a unor operatori cunoscuţi.
  - Cu alte cuvinte, avem o modalitate clară de rezolvare, toate elementele implicate în acest proces sunt specificate.
  - Sunt rezolvate prin metode standard au prin metode cunoscute pentru probleme similare sau analoage.
- Exemplu: rezolvarea unei ecuaţii algebrice de gradul întâi: $5x = 12$.
  - Ştim că pentru problema generală: $a \cdot x = b$, soluţia este: $x = \frac{b}{a}$, iar împărţirea este o operaţie cunoscută cu restricţia $a \neq 0$.
  - Deci în cazul nostru avem: $x = \frac{12}{5} = 2.4$.

Rezolvarea problemelor

- Problemele bine definite
  - Se caracterizează prin existenţa unui scop explicit şi a unor operatori cunoscuţi.
  - Cu alte cuvinte, avem o modalitate clară de rezolvare, toate elementele implicate în acest proces sunt specificate.
  - Sunt rezolvate prin metode standard au prin metode cunoscute pentru probleme similare sau analoage.
- Exemplu: rezolvarea unei ecuaţii algebrice de gradul întâi: $5x = 12$.
  - Ştim că pentru problema generală: $a \cdot x = b$, soluţia este: $x = \frac{b}{a}$, iar împărţirea este o operaţie cunoscută cu restricţia $a \neq 0$.
  - Deci în cazul nostru avem: $x = \frac{12}{5} = 2.4$.

Rezolvarea problemelor

- Clasa problemelor bine definite cuprinde şi problemele dificile "grele" (NP-hard) care necesită un timp de rezolvare non-polinomial şi care devine mult prea mare atunci când dimensiunea problemei creşte.
- În acest caz, pentru a determina totuşi o soluţie, avem nevoie de o modalitate de aproximare a soluţiei optime:
- Algoritmi euristici se bazează pe metode de rezolvare rezultate mai mult din experienţa practică, care încearcă să estimeze o soluţie admisibilă, eficientă din punct de vedere computaţional, cât mai apropiată de soluţia optimă.
- Algoritmi meta-euristici. "A metaheuristic is defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space; learning strategies are used to structure information to find efficiently near-optimal solutions".
- Euristicile sunt specifice problemelor de rezolvat; metaeuristicile sunt proceduri generice care pot fi uşor adaptate pentru diferite clase de probleme

Rezolvarea problemelor

- Clasa problemelor bine definite cuprinde şi problemele dificile "grele" (NP-hard) care necesită un timp de rezolvare non-polinomial şi care devine mult prea mare atunci când dimensiunea problemei creşte.
- În acest caz, pentru a determina totuşi o soluţie, avem nevoie de o modalitate de aproximare a soluţiei optime:
- Algoritmi euristici se bazează pe metode de rezolvare rezultate mai mult din experienţa practică, care încearcă să estimeze o soluţie admisibilă, eficientă din punct de vedere computaţional, cât mai apropiată de soluţia optimă.
- Algoritmi meta-euristici. "A metaheuristic is defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space; learning strategies are used to structure information to find efficiently near-optimal solutions".
- Euristicile sunt specifice problemelor de rezolvat; metaeuristicile sunt proceduri generice care pot fi uşor adaptate pentru diferite clase de probleme

Rezolvarea problemelor

- Clasa problemelor bine definite cuprinde şi problemele dificile "grele" (NP-hard) care necesită un timp de rezolvare non-polinomial şi care devine mult prea mare atunci când dimensiunea problemei creşte.
- În acest caz, pentru a determina totuşi o soluţie, avem nevoie de o modalitate de aproximare a soluţiei optime:
- Algoritmi euristici se bazează pe metode de rezolvare rezultate mai mult din experienţa practică, care încearcă să estimeze o soluţie admisibilă, eficientă din punct de vedere computaţional, cât mai apropiată de soluţia optimă.
- Algoritmi meta-euristici. "A metaheuristic is defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space; learning strategies are used to structure information to find efficiently near-optimal solutions".
- Euristicile sunt specifice problemelor de rezolvat; metaeuristicile sunt proceduri generice care pot fi uşor adaptate pentru diferite clase de probleme

Rezolvarea problemelor

- Clasa problemelor bine definite cuprinde şi problemele dificile "grele" (NP-hard) care necesită un timp de rezolvare non-polinomial şi care devine mult prea mare atunci când dimensiunea problemei creşte.
- În acest caz, pentru a determina totuşi o soluţie, avem nevoie de o modalitate de aproximare a soluţiei optime:
- Algoritmi euristici se bazează pe metode de rezolvare rezultate mai mult din experienţa practică, care încearcă să estimeze o soluţie admisibilă, eficientă din punct de vedere computaţional, cât mai apropiată de soluţia optimă.
- Algoritmi meta-euristici. "A metaheuristic is defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space; learning strategies are used to structure information to find efficiently near-optimal solutions".
- Euristicile sunt specifice problemelor de rezolvat; metaeuristicile sunt proceduri generice care pot fi uşor adaptate pentru diferite clase de probleme

Rezolvarea problemelor

- Clasa problemelor bine definite cuprinde şi problemele dificile "grele" (NP-hard) care necesită un timp de rezolvare non-polinomial şi care devine mult prea mare atunci când dimensiunea problemei creşte.
- În acest caz, pentru a determina totuşi o soluţie, avem nevoie de o modalitate de aproximare a soluţiei optime:
- Algoritmi euristici se bazează pe metode de rezolvare rezultate mai mult din experienţa practică, care încearcă să estimeze o soluţie admisibilă, eficientă din punct de vedere computaţional, cât mai apropiată de soluţia optimă.
- Algoritmi meta-euristici. "A metaheuristic is defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space; learning strategies are used to structure information to find efficiently near-optimal solutions".
- Euristicile sunt specifice problemelor de rezolvat; metaeuristicile sunt proceduri generice care pot fi uşor adaptate pentru diferite clase de probleme

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Probleme rău definite
  - În acest caz intervine un factor de incertitudine, spaţiul problemei nu este complet definit, iar uneori însuşi scopul problemei este dificil de formalizat.
  - Rolul sistemului care rezolvă problema este de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îşi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *învăţare*.
  - Nu este clar de la început care este rezolvarea şi ce soluţie trebuie obţinută.
  - De multe ori, soluţiile găsite mai pot fi îmbunătăţite şi cel care rezolvă problema decide dacă s-a obţinut o soluţie acceptabilă.
- Exemple:
  - Cum salvăm viaţa unui pacient muşcat de şarpe?
  - Cum se scrie un program?
  - Cum se ia 10 la Inteligenţă artificială?

Rezolvarea problemelor

- Din punct de vedere al complexităţii rezolvării şi al relevanţei răspunsului problemele pot fi clasificate în:
  - Probleme pentru care este esenţială obţinerea unui răspuns exact indiferent de resursele implicate. Acestea necesită utilizarea unor tehnici exacte: metode enumerative, backtracking, branch and bound, branch and cut, etc.
  - Probleme pentru care este preferabil să se obţină un răspuns "aproximativ" folosind resurse rezonabile decât un răspuns exact, dar folosind resurse foarte costisitoare.

Rezolvarea problemelor

- Din punct de vedere al complexităţii rezolvării şi al relevanţei răspunsului problemele pot fi clasificate în:
  - Probleme pentru care este esenţială obţinerea unui răspuns exact indiferent de resursele implicate. Acestea necesită utilizarea unor tehnici exacte: metode enumerative, backtracking, branch and bound, branch and cut, etc.
  - Probleme pentru care este preferabil să se obţină un răspuns "aproximativ" folosind resurse rezonabile decât un răspuns exact, dar folosind resurse foarte costisitoare.

Rezolvarea problemelor

- Din punct de vedere al complexităţii rezolvării şi al relevanţei răspunsului problemele pot fi clasificate în:
  - Probleme pentru care este esenţială obţinerea unui răspuns exact indiferent de resursele implicate. Acestea necesită utilizarea unor tehnici exacte: metode enumerative, backtracking, branch and bound, branch and cut, etc.
  - Probleme pentru care este preferabil să se obţină un răspuns "aproximativ" folosind resurse rezonabile decât un răspuns exact, dar folosind resurse foarte costisitoare.

Complexitatea unei probleme

- Un algoritm rezolva o problemă.
- Adesea pentru a rezolva o aceeaşi problemă putem folosi mai mulţi algoritmi, poate cu complexităţi diferite. Există cel puţin 30 de metode de a sorta un şir de valori, de exemplu!
- Putem vorbi deci de complexitatea unui algoritm, dar şi de complexitatea unei probleme.
- Complexitatea unei probleme este complexitatea celui mai "rapid" algoritm care o poate rezolva.
- În general este extrem de dificil de evaluat complexitatea unei probleme, pentru că rareori putem demonstra că un algoritm este cel mai bun posibil.

Complexitatea unei probleme

- Un algoritm rezolva o problemă.
- Adesea pentru a rezolva o aceeaşi problemă putem folosi mai mulţi algoritmi, poate cu complexităţi diferite. Există cel puţin 30 de metode de a sorta un şir de valori, de exemplu!
- Putem vorbi deci de complexitatea unui algoritm, dar şi de complexitatea unei probleme.
- Complexitatea unei probleme este complexitatea celui mai "rapid" algoritm care o poate rezolva.
- În general este extrem de dificil de evaluat complexitatea unei probleme, pentru că rareori putem demonstra că un algoritm este cel mai bun posibil.

Complexitatea unei probleme

- Un algoritm rezolva o problemă.
- Adesea pentru a rezolva o aceeaşi problemă putem folosi mai mulţi algoritmi, poate cu complexităţi diferite. Există cel puţin 30 de metode de a sorta un şir de valori, de exemplu!
- Putem vorbi deci de complexitatea unui algoritm, dar şi de complexitatea unei probleme.
- Complexitatea unei probleme este complexitatea celui mai "rapid" algoritm care o poate rezolva.
- În general este extrem de dificil de evaluat complexitatea unei probleme, pentru că rareori putem demonstra că un algoritm este cel mai bun posibil.

Complexitatea unei probleme

- Un algoritm rezolva o problemă.
- Adesea pentru a rezolva o aceeaşi problemă putem folosi mai mulţi algoritmi, poate cu complexităţi diferite. Există cel puţin 30 de metode de a sorta un şir de valori, de exemplu!
- Putem vorbi deci de complexitatea unui algoritm, dar şi de complexitatea unei probleme.
- Complexitatea unei probleme este complexitatea celui mai "rapid" algoritm care o poate rezolva.
- În general este extrem de dificil de evaluat complexitatea unei probleme, pentru că rareori putem demonstra că un algoritm este cel mai bun posibil.

Complexitatea unei probleme

- Un algoritm rezolva o problemă.
- Adesea pentru a rezolva o aceeaşi problemă putem folosi mai mulţi algoritmi, poate cu complexităţi diferite. Există cel puţin 30 de metode de a sorta un şir de valori, de exemplu!
- Putem vorbi deci de complexitatea unui algoritm, dar şi de complexitatea unei probleme.
- Complexitatea unei probleme este complexitatea celui mai "rapid" algoritm care o poate rezolva.
- În general este extrem de dificil de evaluat complexitatea unei probleme, pentru că rareori putem demonstra că un algoritm este cel mai bun posibil.

Clasa P

- Unele probleme se pot rezolva, altele nu.
- De exemplu, o problema notorie, a cărei imposibilitate este riguros demonstrată în anii ’30 de către matematicianul englez Alan Turing, este de a decide dacă un program se va opri vreodată pentru o anumită instanţă a datelor de intrare.
- Pe de altă parte, chiar între problemele care pot fi rezolvate, se trage o linie imaginară intre problemele care au rezolvari "rezonabil" de rapide, şi restul problemelor, care se numesc "intratabile".
- Mulţimea tuturor problemelor de decizie (adică a problemelor la care răspunsul este da sau nu) cu complexitate polinomială se notează cu P (de la polinom).
- De exemplu, problema de a găsi dacă o valoare se află într-un vector este în clasa P.

Clasa P

- Unele probleme se pot rezolva, altele nu.
- De exemplu, o problema notorie, a cărei imposibilitate este riguros demonstrată în anii ’30 de câtre matematicianul englez Alan Turing, este de a decide dacă un program se va opri vreodată pentru o anumită instanţă a datelor de intrare.
- Pe de altă parte, chiar între problemele care pot fi rezolvate, se trage o linie imaginară intre problemele care au rezolvari "rezonabil" de rapide, şi restul problemelor, care se numesc "intratabile".
- Mulţimea tuturor problemelor de decizie (adică a problemelor la care răspunsul este da sau nu) cu complexitate polinomială se notează cu P (de la polinom).
- De exemplu, problema de a găsi dacă o valoare se află într-un vector este în clasa P.

Clasa P

- Unele probleme se pot rezolva, altele nu.
- De exemplu, o problema notorie, a cărei imposibilitate este riguros demonstrată în anii ’30 de către matematicianul englez Alan Turing, este de a decide dacă un program se va opri vreodată pentru o anumită instanţă a datelor de intrare.
- Pe de altă parte, chiar între problemele care pot fi rezolvate, se trage o linie imaginară intre problemele care au rezolvari "rezonabil" de rapide, şi restul problemelor, care se numesc "intratabile".
- Mulţimea tuturor problemelor de decizie (adică a problemelor la care răspunsul este da sau nu) cu complexitate polinomială se notează cu P (de la polinom).
- De exemplu, problema de a găsi dacă o valoare se află într-un vector este în clasa P.

Clasa P

- Unele probleme se pot rezolva, altele nu.
- De exemplu, o problema notorie, a cărei imposibilitate este riguros demonstrată în anii ’30 de către matematicianul englez Alan Turing, este de a decide dacă un program se va opri vreodată pentru o anumită instanţă a datelor de intrare.
- Pe de altă parte, chiar între problemele care pot fi rezolvate, se trage o linie imaginară intre problemele care au rezolvari "rezonabil" de rapide, şi restul problemelor, care se numesc "intratabile".
- Mulţimea tuturor problemelor de decizie (adică a problemelor la care răspunsul este da sau nu) cu complexitate polinomială se notează cu P (de la polinom).
- De exemplu, problema de a găsi dacă o valoare se află într-un vector este în clasa P.

Clasa P

- Unele probleme se pot rezolva, altele nu.
- De exemplu, o problema notorie, a cărei imposibilitate este riguros demonstrată în anii ’30 de către matematicianul englez Alan Turing, este de a decide dacă un program se va opri vreodată pentru o anumită instanţă a datelor de intrare.
- Pe de altă parte, chiar între problemele care pot fi rezolvate, se trage o linie imaginară intre problemele care au rezolvari "rezonabil" de rapide, şi restul problemelor, care se numesc "intratabile".
- Mulţimea tuturor problemelor de decizie (adică a problemelor la care răspunsul este da sau nu) cu complexitate polinomială se notează cu P (de la polinom).
- De exemplu, problema de a găsi dacă o valoare se află într-un vector este în clasa P.

Clasa P

Definiţie. În teoria computabilităţii şi teoria complexităţii computaţionale, o **problemă de decizie** este o problemă care poate fi pusă ca o întrebare da-nu a valorilor de intrare.

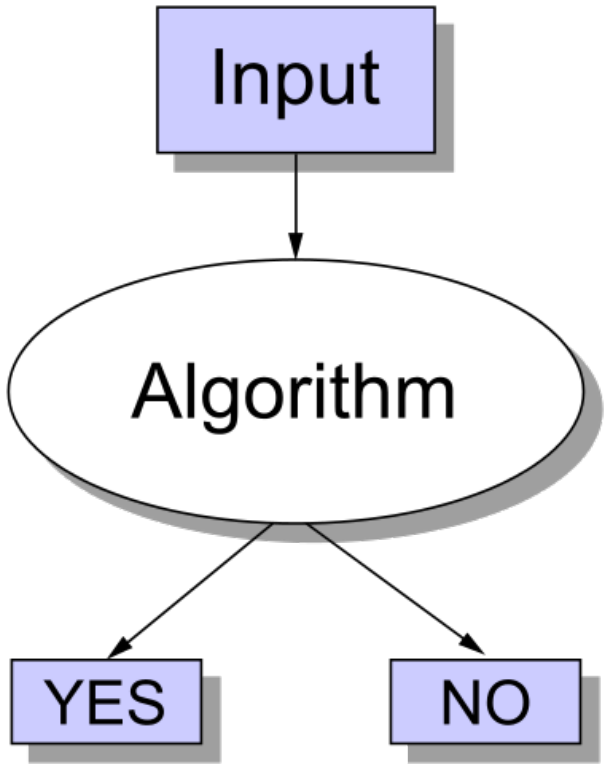


Figure: O problemă de decizie are doar două ieşiri (outputs) posibile (da sau nu) pe orice intrare (input).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Algoritmii cu care suntem obişnuiţi să lucrăm sunt deterministi (la un moment dat evoluţia algoritmului este unic determinată, şi deasemenea instrucţiunea care urmează să se execute este unic precizată în fiecare moment).
- Un algoritm nedeterminist este un algoritm în care este permisă trecerea (necondiţionată) dintr-o stare dată in mai multe stări următoare şi care poate efectua *simultan* mai multe calcule independente.
- Un algoritm nedeterminist este corect dacă există o posibilitate de executare a sa care găseşte răspunsul corect.
- Clasa tuturor problemelor care se pot rezolva cu algoritmi nedeterminişti într-un timp de execuţie polinomial se notează cu NP (Nedeterminist Polinomial).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Algoritmii cu care suntem obişnuiţi să lucrăm sunt deterministi (la un moment dat evoluţia algoritmului este unic determinată, şi deasemenea instrucţiunea care urmează să se execute este unic precizată în fiecare moment).
- Un algoritm nedeterminist este un algoritm în care este permisă trecerea (necondiţionată) dintr-o stare dată in mai multe stări următoare şi care poate efectua *simultan* mai multe calcule independente.
- Un algoritm nedeterminist este corect dacă există o posibilitate de executare a sa care găseşte răspunsul corect.
- Clasa tuturor problemelor care se pot rezolva cu algoritmi nedeterminişti într-un timp de execuţie polinomial se notează cu NP (Nedeterminist Polinomial).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Algoritmii cu care suntem obişnuiţi să lucrăm sunt deterministi (la un moment dat evoluţia algoritmului este unic determinată, şi deasemenea instrucţiunea care urmează să se execute este unic precizată în fiecare moment).
- Un algoritm nedeterminist este un algoritm în care este permisă trecerea (necondiţionată) dintr-o stare dată in mai multe stări următoare şi care poate efectua *simultan* mai multe calcule independente.
- Un algoritm nedeterminist este corect dacă există o posibilitate de executare a sa care găseşte răspunsul corect.
- Clasa tuturor problemelor care se pot rezolva cu algoritmi nedeterminişti într-un timp de execuţie polinomial se notează cu NP (Nedeterminist Polinomial).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Algoritmii cu care suntem obişnuiţi să lucrăm sunt deterministi (la un moment dat evoluţia algoritmului este unic determinată, şi deasemenea instrucţiunea care urmează să se execute este unic precizată în fiecare moment).
- Un algoritm nedeterminist este un algoritm în care este permisă trecerea (necondiţionată) dintr-o stare dată in mai multe stări următoare şi care poate efectua *simultan* mai multe calcule independente.
- Un algoritm nedeterminist este corect dacă există o posibilitate de executare a sa care găseşte răspunsul corect.
- Clasa tuturor problemelor care se pot rezolva cu algoritmi nedeterminişti într-un timp de execuţie polinomial se notează cu NP (Nedeterminist Polinomial).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Este clar că orice problemă care se află in P se află şi in NP, pentru că algoritmii determinişti sunt doar un caz extrem al celor determinişti: în fiecare moment au o singură alegere posibilă.
- Din păcate transformarea într-un algoritm determinist se face pierzând din eficienţă.
- În general un algoritm care operează în timp nedeterminist polinomial (NP) poate fi transformat cu uşurinţă într-un algoritm cu timp exponenţial (EXP).
- Avem deci următoarea incluziune de mulţimi între problemele de decizie:

$$P \subseteq NP \subseteq EXP$$

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Este clar că orice problemă care se află in P se află şi in NP, pentru că algoritmii determinişti sunt doar un caz extrem al celor determinişti: în fiecare moment au o singură alegere posibilă.
- Din păcate transformarea într-un algoritm determinist se face pierzând din eficienţă.
- În general un algoritm care operează în timp nedeterminist polinomial (NP) poate fi transformat cu uşurinţă într-un algoritm cu timp exponenţial (EXP).
- Avem deci următoarea incluziune de mulţimi între problemele de decizie:

$$P \subseteq NP \subseteq EXP$$

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Este clar că orice problemă care se află in *P* se află şi in *NP*, pentru că algoritmii determinişti sunt doar un caz extrem al celor determinişti: în fiecare moment au o singură alegere posibilă.
- Din păcate transformarea într-un algoritm determinist se face pierzând din eficienţă.
- În general un algoritm care operează în timp nedeterminist polinomial (*NP*) poate fi transformat cu uşurinţă într-un algoritm cu timp exponenţial (*EXP*).
- Avem deci următoarea incluziune de mulţimi între problemele de decizie:

$$P \subseteq NP \subseteq EXP$$

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Este clar că orice problemă care se află in P se află şi in NP, pentru că algoritmii determinişti sunt doar un caz extrem al celor determinişti: în fiecare moment au o singură alegere posibilă.
- Din păcate transformarea într-un algoritm determinist se face pierzând din eficienţă.
- În general un algoritm care operează în timp nedeterminist polinomial (NP) poate fi transformat cu uşurinţă într-un algoritm cu timp exponenţial (EXP).
- Avem deci următoarea incluziune de mulţimi între problemele de decizie:

$$P \subseteq NP \subseteq EXP$$

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Partea cea mai interesantă este următoarea: ştim cu certitudine că $P \neq EXP$.
- Însă nu avem nici o idee despre relaţia de egalitate între NP şi P sau între NP şi EXP.
- Nu există nici o demonstraţie care să infirme că problemele din NP au algoritmi eficienţi, determinist polinomiali!
- Problema $P = NP$ este cea mai importantă problemă din teoria calculatoarelor, pentru că de soluţionarea ei se leagă o serie de consecinţe importante.
- In 1971 Cook a demonstrat că există o problemă specială în NP (adică pentru care se poate da un algoritm eficient nedeterminist), numită problema satisfiabilităţii (notată cu SAT).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Partea cea mai interesantă este următoarea: ştim cu certitudine că $P \neq EXP$.
- Însă nu avem nici o idee despre relaţia de egalitate între NP şi P sau între NP şi EXP.
- Nu există nici o demonstraţie care să infirme că problemele din NP au algoritmi eficienţi, determinist polinomiali!
- Problema $P = NP$ este cea mai importantă problemă din teoria calculatoarelor, pentru că de soluţionarea ei se leagă o serie de consecinţe importante.
- In 1971 Cook a demonstrat că există o problemă specială în NP (adică pentru care se poate da un algoritm eficient nedeterminist), numită problema satisfiabilităţii (notată cu SAT).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Partea cea mai interesantă este următoarea: ştim cu certitudine că $P \neq EXP$.
- Însă nu avem nici o idee despre relaţia de egalitate între NP şi P sau între NP şi EXP.
- Nu există nici o demonstraţie care să infirme că problemele din NP au algoritmi eficienţi, determinist polinomiali!
- Problema $P = NP$ este cea mai importantă problemă din teoria calculatoarelor, pentru că de soluţionarea ei se leagă o serie de consecinţe importante.
- In 1971 Cook a demonstrat că există o problemă specială în NP (adică pentru care se poate da un algoritm eficient nedeterminist), numită problema satisfiabilităţii (notată cu SAT).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Partea cea mai interesantă este următoarea: ştim cu certitudine că $P \neq EXP$.
- Însă nu avem nici o idee despre relaţia de egalitate între NP şi P sau între NP şi EXP.
- Nu există nici o demonstraţie care să infirme că problemele din NP au algoritmi eficienţi, determinist polinomiali!
- Problema $P = NP$ este cea mai importantă problemă din teoria calculatoarelor, pentru că de soluţionarea ei se leagă o serie de consecinţe importante.
- In 1971 Cook a demonstrat că există o problemă specială în NP (adică pentru care se poate da un algoritm eficient nedeterminist), numită problema satisfiabilităţii (notată cu SAT).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Partea cea mai interesantă este următoarea: ştim cu certitudine că $P \neq EXP$.
- Însă nu avem nici o idee despre relaţia de egalitate între NP şi P sau între NP şi EXP.
- Nu există nici o demonstraţie care să infirme că problemele din NP au algoritmi eficienţi, determinist polinomiali!
- Problema $P = NP$ este cea mai importantă problemă din teoria calculatoarelor, pentru că de soluţionarea ei se leagă o serie de consecinţe importante.
- In 1971 Cook a demonstrat că există o problemă specială în NP (adică pentru care se poate da un algoritm eficient nedeterminist), numită problema satisfiabilităţii (notată cu SAT).

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- O problemă P_1 se reduce polinomial la o problemă P_2 dacă orice caz particular al problemei P_2 se poate transforma în timp polinomial într-un caz particular al problemei P_1 şi dacă soluţia problemei P_2 se poate obţine în timp polinomial din soluţia problemei P_1 ($P_2 \rightarrow P_1$).
- O problemă este NP-dificilă dacă orice problemă din clasa NP se reduce la ea.
- O problemă este NP-completă dacă este NP-dificilă şi clasei NP.
- Exemple de probleme NP-complete:
  - Problema comis-voiajorului: dându-se o reţea de oraşe, o reţea de drumuri între oraşe şi o lungime k, există un traseu de cost mai mic decât k trecând prin fiecare oraş o singură dată si revenind la punctul de plecare?

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- O problemă P_1 se reduce polinomial la o problemă P_2 dacă orice caz particular al problemei P_2 se poate transforma în timp polinomial într-un caz particular al problemei P_1 şi dacă soluţia problemei P_2 se poate obţine în timp polinomial din soluţia problemei P_1 ($P_2 \rightarrow P_1$).
- O problemă este NP-dificilă dacă orice problemă din clasa NP se reduce la ea.
- O problemă este NP-completă dacă este NP-dificilă şi clasei NP.
- Exemple de probleme NP-complete:
  - Problema comis-voiajorului: dându-se o reţea de oraşe, o reţea de drumuri între oraşe şi o lungime k, există un traseu de cost mai mic decât k trecând prin fiecare oraş o singură dată si revenind la punctul de plecare?

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- O problemă P_1 se reduce polinomial la o problemă P_2 dacă orice caz particular al problemei P_2 se poate transforma în timp polinomial într-un caz particular al problemei P_1 şi dacă soluţia problemei P_2 se poate obţine în timp polinomial din soluţia problemei P_1 ($P_2 \rightarrow P_1$).
- O problemă este NP-dificilă dacă orice problemă din clasa NP se reduce la ea.
- O problemă este NP-completă dacă este NP-dificilă şi clasei NP.
- Exemple de probleme NP-complete:
  - Problema comis-voiajorului: dându-se o reţea de oraşe, o reţea de drumuri între oraşe şi o lungime k, există un traseu de cost mai mic decât k trecând prin fiecare oraş o singură dată si revenind la punctul de plecare?

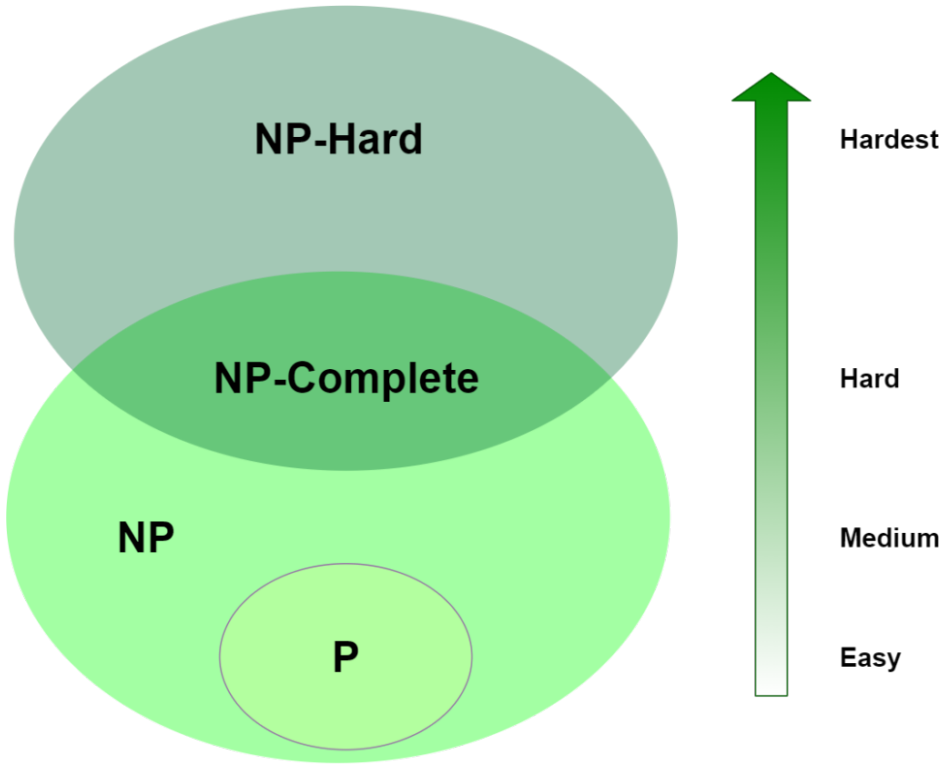
Clasa NP; algoritmi nedeterminişti; NP-completitudine

- O problemă P_1 se reduce polinomial la o problemă P_2 dacă orice caz particular al problemei P_2 se poate transforma în timp polinomial într-un caz particular al problemei P_1 şi dacă soluţia problemei P_2 se poate obţine în timp polinomial din soluţia problemei P_1 ($P_2 \rightarrow P_1$).
- O problemă este NP-dificilă dacă orice problemă din clasa NP se reduce la ea.
- O problemă este NP-completă dacă este NP-dificilă şi clasei NP.
- Exemple de probleme NP-complete:
  - Problema comis-voiajorului: dându-se o reţea de oraşe, o reţea de drumuri între oraşe şi o lungime k, există un traseu de cost mai mic decât k trecând prin fiecare oraş o singură dată si revenind la punctul de plecare?

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- O problemă P_1 se reduce polinomial la o problemă P_2 dacă orice caz particular al problemei P_2 se poate transforma în timp polinomial într-un caz particular al problemei P_1 şi dacă soluţia problemei P_2 se poate obţine în timp polinomial din soluţia problemei P_1 ($P_2 \rightarrow P_1$).
- O problemă este NP-dificilă dacă orice problemă din clasa NP se reduce la ea.
- O problemă este NP-completă dacă este NP-dificilă şi clasei NP.
- Exemple de probleme NP-complete:
  - Problema comis-voiajorului: dându-se o reţea de oraşe, o reţea de drumuri între oraşe şi o lungime k, există un traseu de cost mai mic decât k trecând prin fiecare oraş o singură dată si revenind la punctul de plecare?

Clasa NP; algoritmi nedeterminişti; NP-completitudine



Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Alte exemple de probleme NP-complete:
  - Clica: dându-se un graf G şi un număr k, are G un subgraf complet cu k vârfuri (adică o mulţime de k vârfuri unite fiecare cu fiecare)?
  - Acoperire: dându-se un graf G şi un număr k, pot alege k vărfuri în aşa fel încât toate muchiile din G au un capăt ales?
- O cantitate enormă de efort şi ingeniozitate a fost risipită pentru a încerca să se demonstreze că $P = NP$ sau opusul acestei afirmaţii, dar nici un rezultat concret nu a fost obţinut.
- Credinţa cvasi-unanimă este că $P \neq NP$, dar numai matematica şi informatica pot oferi vreo certitudine...

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Alte exemple de probleme NP-complete:
  - Clica: dându-se un graf G şi un număr k, are G un subgraf complet cu k vârfuri (adică o mulţime de k vârfuri unite fiecare cu fiecare)?
  - Acoperire: dându-se un graf G şi un număr k, pot alege k vărfuri în aşa fel încât toate muchiile din G au un capăt ales?
- O cantitate enormă de efort şi ingeniozitate a fost risipită pentru a încerca să se demonstreze că $P = NP$ sau opusul acestei afirmaţii, dar nici un rezultat concret nu a fost obţinut.
- Credinţa cvasi-unanimă este că $P \neq NP$, dar numai matematica şi informatica pot oferi vreo certitudine...

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Alte exemple de probleme NP-complete:
  - Clica: dându-se un graf G şi un număr k, are G un subgraf complet cu k vârfuri (adică o mulţime de k vârfuri unite fiecare cu fiecare)?
  - Acoperire: dându-se un graf G şi un număr k, pot alege k vărfuri în aşa fel încât toate muchiile din G au un capăt ales?
- O cantitate enormă de efort şi ingeniozitate a fost risipită pentru a încerca să se demonstreze că $P = NP$ sau opusul acestei afirmaţii, dar nici un rezultat concret nu a fost obţinut.
- Credinţa cvasi-unanimă este că $P \neq NP$, dar numai matematica şi informatica pot oferi vreo certitudine...

Clasa NP; algoritmi nedeterminişti; NP-completitudine

- Alte exemple de probleme NP-complete:
  - Clica: dându-se un graf G şi un număr k, are G un subgraf complet cu k vârfuri (adică o mulţime de k vârfuri unite fiecare cu fiecare)?
  - Acoperire: dându-se un graf G şi un număr k, pot alege k vărfuri în aşa fel încât toate muchiile din G au un capăt ales?
- O cantitate enormă de efort şi ingeniozitate a fost risipită pentru a încerca să se demonstreze că $P = NP$ sau opusul acestei afirmaţii, dar nici un rezultat concret nu a fost obţinut.
- Credinţa cvasi-unanimă este că $P \neq NP$, dar numai matematica şi informatica pot oferi vreo certitudine...

Calcul inteligent

- Calculul inteligent este un domeniu al Inteligenţei Artificiale care grupează tehnici de rezolvare a problemelor dificile din categoria NP-hard.
- Principalele direcţii ale calculului inteligent care vor fi investigate pe parcursul acestui curs sunt:
  - Calculul neuronal. Este folosit în general la rezolvarea problemelor de asociere, bazându-se pe extragerea, prin învăţare, a unui model pornind de la exemple. Sursa de inspiraţie o reprezintă structura şi funcţionarea creierului uman.
  - Calculul evolutiv. Este folosit în principal în rezolvarea problemelor bazate pe căutarea soluţiei într-un spaţiu mare de soluţii potenţiale. Sursa de inspiraţie o reprezintă principiile evoluţionismului darwinist.
  - Calculul fuzzy. Este folosit atunci când datele problemei nu pot fi descrise exact ci se caracterizează prin prezenţa unui grad de incertitudine. Ideea de bază este de a înlocui valorile exacte cu valori "fuzzy" descrise prin funcţii de apartenenţă.

Calcul inteligent

- Calculul inteligent este un domeniu al Inteligenţei Artificiale care grupează tehnici de rezolvare a problemelor dificile din categoria NP-hard.
- Principalele direcţii ale calculului inteligent care vor fi investigate pe parcursul acestui curs sunt:
  - Calculul neuronal. Este folosit în general la rezolvarea problemelor de asociere, bazându-se pe extragerea, prin învăţare, a unui model pornind de la exemple. Sursa de inspiraţie o reprezintă structura şi funcţionarea creierului uman.
  - Calculul evolutiv. Este folosit în principal în rezolvarea problemelor bazate pe căutarea soluţiei într-un spaţiu mare de soluţii potenţiale. Sursa de inspiraţie o reprezintă principiile evoluţionismului darwinist.
  - Calculul fuzzy. Este folosit atunci când datele problemei nu pot fi descrise exact ci se caracterizează prin prezenţa unui grad de incertitudine. Ideea de bază este de a înlocui valorile exacte cu valori "fuzzy" descrise prin funcţii de apartenenţă.

Calcul inteligent

- Calculul inteligent este un domeniu al Inteligenţei Artificiale care grupează tehnici de rezolvare a problemelor dificile din categoria NP-hard.
- Principalele direcţii ale calculului inteligent care vor fi investigate pe parcursul acestui curs sunt:
  - Calculul neuronal. Este folosit în general la rezolvarea problemelor de asociere, bazându-se pe extragerea, prin învăţare, a unui model pornind de la exemple. Sursa de inspiraţie o reprezintă structura şi funcţionarea creierului uman.
  - Calculul evolutiv. Este folosit în principal în rezolvarea problemelor bazate pe căutarea soluţiei într-un spaţiu mare de soluţii potenţiale. Sursa de inspiraţie o reprezintă principiile evoluţionismului darwinist.
  - Calculul fuzzy. Este folosit atunci când datele problemei nu pot fi descrise exact ci se caracterizează prin prezenţa unui grad de incertitudine. Ideea de bază este de a înlocui valorile exacte cu valori "fuzzy" descrise prin funcţii de apartenenţă.

Calcul inteligent

- Calculul inteligent este un domeniu al Inteligenţei Artificiale care grupează tehnici de rezolvare a problemelor dificile din categoria NP-hard.
- Principalele direcţii ale calculului inteligent care vor fi investigate pe parcursul acestui curs sunt:
  - Calculul neuronal. Este folosit în general la rezolvarea problemelor de asociere, bazându-se pe extragerea, prin învăţare, a unui model pornind de la exemple. Sursa de inspiraţie o reprezintă structura şi funcţionarea creierului uman.
  - Calculul evolutiv. Este folosit în principal în rezolvarea problemelor bazate pe căutarea soluţiei într-un spaţiu mare de soluţii potenţiale. Sursa de inspiraţie o reprezintă principiile evoluţionismului darwinist.
  - Calculul fuzzy. Este folosit atunci când datele problemei nu pot fi descrise exact ci se caracterizează prin prezenţa unui grad de incertitudine. Ideea de bază este de a înlocui valorile exacte cu valori "fuzzy" descrise prin funcţii de apartenenţă.

Calcul inteligent

- Calculul inteligent este un domeniu al Inteligenţei Artificiale care grupează tehnici de rezolvare a problemelor dificile din categoria NP-hard.
- Principalele direcţii ale calculului inteligent care vor fi investigate pe parcursul acestui curs sunt:
  - Calculul neuronal. Este folosit în general la rezolvarea problemelor de asociere, bazându-se pe extragerea, prin învăţare, a unui model pornind de la exemple. Sursa de inspiraţie o reprezintă structura şi funcţionarea creierului uman.
  - Calculul evolutiv. Este folosit în principal în rezolvarea problemelor bazate pe căutarea soluţiei într-un spaţiu mare de soluţii potenţiale. Sursa de inspiraţie o reprezintă principiile evoluţionismului darwinist.
  - Calculul fuzzy. Este folosit atunci când datele problemei nu pot fi descrise exact ci se caracterizează prin prezenţa unui grad de incertitudine. Ideea de bază este de a înlocui valorile exacte cu valori "fuzzy" descrise prin funcţii de apartenenţă.

Calcul inteligent

- În fiecare dintre cele trei direcţii majoritatea operaţiilor care se efectuează au un caracter numeric, fiind necesară o codificare numerică adecvată a problemei. Aceasta motivează existenţa cuvântului *calcul* în denumirea domeniului.
- Pe de altă parte în fiecare dintre direcţiile prezentate se încearcă simularea unor comportamente inteligente ceea ce motivează prezenţa termenului *inteligent*.
- Principiul fundamental al calculului neuronal şi al celui evolutiv este de a dezvolta sisteme de calcul inteligent pornind de la implementarea unor reguli simple, comportamentu complex al acestor sisteme derivând din aplicarea în paralel şi în manieră interactivă a acestor reguli.
- Această abordare de tip "bottom-up" este în contrast cu abordarea de tip "top-down" specifică celorlalte domenii ale Inteligenţei Artificiale.

Calcul inteligent

- În fiecare dintre cele trei direcţii majoritatea operaţiilor care se efectuează au un caracter numeric, fiind necesară o codificare numerică adecvată a problemei. Aceasta motivează existenţa cuvântului *calcul* în denumirea domeniului.
- Pe de altă parte în fiecare dintre direcţiile prezentate se încearcă simularea unor comportamente inteligente ceea ce motivează prezenţa termenului *inteligent*.
- Principiul fundamental al calculului neuronal şi al celui evolutiv este de a dezvolta sisteme de calcul inteligent pornind de la implementarea unor reguli simple, comportamentu complex al acestor sisteme derivând din aplicarea în paralel şi în manieră interactivă a acestor reguli.
- Această abordare de tip "bottom-up" este în contrast cu abordarea de tip "top-down" specifică celorlalte domenii ale Inteligenţei Artificiale.

Calcul inteligent

- În fiecare dintre cele trei direcţii majoritatea operaţiilor care se efectuează au un caracter numeric, fiind necesară o codificare numerică adecvată a problemei. Aceasta motivează existenţa cuvântului *calcul* în denumirea domeniului.
- Pe de altă parte în fiecare dintre direcţiile prezentate se încearcă simularea unor comportamente inteligente ceea ce motivează prezenţa termenului *inteligent*.
- Principiul fundamental al calculului neuronal şi al celui evolutiv este de a dezvolta sisteme de calcul inteligent pornind de la implementarea unor reguli simple, comportamentu complex al acestor sisteme derivând din aplicarea în paralel şi în manieră interactivă a acestor reguli.
- Această abordare de tip "bottom-up" este în contrast cu abordarea de tip "top-down" specifică celorlalte domenii ale Inteligenţei Artificiale.

Calcul inteligent

- În fiecare dintre cele trei direcţii majoritatea operaţiilor care se efectuează au un caracter numeric, fiind necesară o codificare numerică adecvată a problemei. Aceasta motivează existenţa cuvântului *calcul* în denumirea domeniului.
- Pe de altă parte în fiecare dintre direcţiile prezentate se încearcă simularea unor comportamente inteligente ceea ce motivează prezenţa termenului *inteligent*.
- Principiul fundamental al calculului neuronal şi al celui evolutiv este de a dezvolta sisteme de calcul inteligent pornind de la implementarea unor reguli simple, comportamentu complex al acestor sisteme derivând din aplicarea în paralel şi în manieră interactivă a acestor reguli.
- Această abordare de tip "bottom-up" este în contrast cu abordarea de tip "top-down" specifică celorlalte domenii ale Inteligenţei Artificiale.

For Further Reading I

- Alexander Schrijver, A Course in Combinatorial Optimization, February 1, 2006.
- William J. Cook, William H. Cunningham, William R. Pulleyblank, Alexander Schrijver, Combinatorial Optimization; John Wiley & Sons; 1 edition (November 12, 1997).
- Jon Lee, A First Course in Combinatorial Optimization; Cambridge University Press; 2004.
- Pierluigi Crescenzi, Viggo Kann, Magnús Halldórsson, Marek Karpinski, Gerhard Woeginger, A Compendium of NP Optimization Problems.
- Christos H. Papadimitriou and Kenneth Steiglitz, Combinatorial Optimization : Algorithms and Complexity; Dover Pubns; (paperback, Unabridged edition, July 1998).