POLSKA AKADEMIA NAUK
Instytut Badań Systemowych

ANALIZA PRZYPADKU ŚREDNIEGO W OPTYMALIZACJI DYSKRETNEJ

Wielowymiarowe zadanie załadunku oraz zadanie szeregowania prac

Krzysztof Szkatuła

Krzysztof SZKATUŁA - ANALIZA PRZYPADKU ŚREDNIEGO W OPTYMALIZACJI DYSKRETNEJ

ANALIZA PRZYPADKU ŚREDNIEGO W OPTYMALIZACJI DYSKRETNEJ

Polska Akademia Nauk • Instytut Badań Systemowych

Seria: BADANIA SYSTEMOWE
tom 23

Redaktor naukowy:

Prof. dr hab. Jakub Gutenbaum

Warszawa 1999

Krzysztof SZKATUŁA

ANALIZA PRZYPADKU ŚREDNIEGO W OPTYMALIZACJI DYSKRETNEJ

Wielowymiarowe zadanie załadunku oraz zadanie szeregowania prac

Publikację opiniowali do druku:

Prof. dr hab. Juliusz L. Kulikowski
Dr hab. Włodzimierz Ogryczak

ISBN 83-85847-39-1
ISSN 0208-8029

Wprowadzenie

Optymalizacja dyskretna stała się samodzielną dziedziną badawczą od połowy lat pięćdziesiątych dwudziestego wieku. Powstała ona na styku zastosowań praktycznych w dziedzinach takich jak ekonomia, zarządzanie, technika i wiele innych oraz matematyki ze szczególnym uwzględnieniem kombinatoryki, teorii grafów i logiki matematycznej. W pewnym uproszczeniu można powiedzieć, że podstawowym celem optymalizacji dyskretnej jest wybór optymalnego wariantu ze skończonego lub przeliczalnego ich zbioru. Optymalność jest rozumiana jako wyznaczenie maksimum lub minimum pewnej funkcji. Uzyskanie rozwiązania zadania optymalizacji dyskretnej umożliwia podejmowanie trafnych decyzji w odniesieniu do wielu aspektów działalności ludzkiej. Przykładami kryteriów optymalizacyjnych może być maksymalizacja zysków, minimalizacja kosztów lub strat i wiele innych.

Można powiedzieć, że w zaawansowanych zastosowaniach praktycznych pojawiło się wiele ważnych, złożonych i trudnych do rozwiązania problemów o powyższym charakterze. Do ich sformalizowania w najbardziej odpowiedni sposób bardzo przydatny okazał się aparat i metodologia matematyki.

W momencie gdy zagadnienie praktyczne zostało sformalizowane jako optymalizacyjny problem matematyczny, powstaje potrzeba jego efektywnego rozwiązania. Oznacza to konieczność opracowania algorytmów i wykonania ich komputerowej implementacji. Niestety, okazało się, że w przypadku wielu zadań optymalizacji dyskretnej oraz algorytmów opracowanych do ich rozwiązywania pojawia się zasadnicza trudność. Dla nawet niezbyt dużych przykładów tych zadań obliczenia numeryczne wymagają niemożliwego do zaakceptowania nakładu obliczeń, na przykład mierzonego w stuleciach pracy obecnych superkomputerów. Co więcej, nawet drastyczne zwiększenie wydajności komputerów nie jest w stanie zasadniczo poprawić sytuacji. Dla ustalenia uwagi, 100-krotne przyśpieszenie obliczeń zmniejsza nakład obliczeń ze 100 lat do jednego roku, co wciąż jest wielkością abstrakcyjną, niemożliwą do zaakceptowania w praktyce obliczeniowej. Efektem tej sytuacji był rozwój teorii i praktycznego zastosowania algorytmów przybliżonych, których celem jest wyznaczenie przybliżonego rozwiązania zadania o akceptowalnej jakości,

przy "niewielkim" nakładzie obliczeń.

Praktyczna niemożliwość uzyskania rozwiązań optymalnych dla licznych przykładów zadań optymalizacji dyskretnej spowodowała konieczność analizowania nakładu obliczeń wymaganego przez algorytmy, dla zadań o określonej wielkości.

W efekcie powstała dziedzina badawcza zwana złożonością obliczeniową. Jej podstawowym zadaniem jest analizowanie zadań i algorytmów optymalizacji dyskretnej z punktu widzenia oceny nakładu obliczeń niezbędnego do uzyskania rozwiązania optymalnego jako funkcji rozmiaru, wielkości zadania. Zadania optymalizacji dyskretnej zostały umownie podzielone na łatwe i trudne do rozwiązywania.

Należy zwrócić uwagę na fakt, że uzyskane w ten sposób oceny noszą charakter absolutnej gwarancji, to znaczy, że są one prawdziwe dla wszystkich realizacji danych analizowanego zadania. Analiza taka nosi nazwę analizy przypadku najgorszego, gdyż oparta jest ona na najbardziej niekorzystnym zachowaniu się zadań i algorytmów optymalizacji dyskretnej.

Praktyka rozwiązywania zadań optymalizacji dyskretnej wykazała jednak szybko, że uzyskane w ten sposób oceny są bardzo często nadmiernie pesymistyczne. Oceny uzyskane w oparciu o podejście przypadku najgorszego nie charakteryzują w sposób właściwy przeciętnego, średniego zachowania się zadań i algorytmów optymalizacji dyskretnej.

Ta sytuacja spowodowała powstanie dziedziny nazywanej analizą przypadku średniego lub inaczej analizą probabilistyczną zadań i algorytmów optymalizacji dyskretnej. Przeprowadzenie analizy przypadku średniego wymaga zdefniowania losowego modelu rozważanego zadania. Uzyskane w efekcie przeprowadzenia analizy przypadku średniego wyniki odnoszą się tylko do rozważanego losowego modelu zadania optymalizacji dyskretnej. Natomiast ich olbrzymią zaletą jest możliwość uzyskania alternatywnych, w stosunku do złożoności obliczeniowej w przypadku najgorszym, ocen zachowania się zadań i algorytmów optymalizacji dyskretnej.

Ważną i oryginalną właściwością analizy przypadku średniego jest możliwość analizowania innych charakterystyk zadania niż złożoność obliczeniowa. Dobrym przykładem jest asymptotyczna ocena zachowania się wartości rozwiązania optymalnego jako funkcji pewnych parametrów zadania. Wyniki tego typu znacznie poszerzają wiedzę na temat zadań optymalizacji dyskretnej i w efekcie umożliwiają ich rozwiązywanie w znacznie bardziej efektywny sposób.

Zasadniczym celem tej monografii jest wykazanie, na przykładzie wybranych, ważnych zadań optymalizacji dyskretnej, że przy zastosowaniu prostego aparatu rachunku prawdopodobieństwa można uzyskać wartościowe wyniki analizy przypadku średniego.

W celu właściwego osadzenia uzyskanych wyników w teorii i praktyce optymalizacji dyskretnej pierwsze rozdziały monografii poświęcone zostały prezentacji wybranych zadań optymalizacji dyskretnej, metod ich rozwiązywania, złożoności obliczeniowej oraz analizy przypadku średniego. Należy jednak podkreślić, że zamiarem autora była poglądowa, a nie bardzo szczegółowa i wyczerpująca prezentacja tych zagadnień. Szczegółowy plan monografii jest następujący.

W rozdziale 1 zaprezentowano dziedzinę optymalizacji dyskretnej ze szczególnym uwzględnieniem programowania całkowitoliczbowego i liniowego, zadań binarnych, zadań teorii grafów oraz zadań harmonogramowania.

W rozdziale 2 zostały przedstawione znane metody rozwiązywania zadań optymalizacji dyskretnej, takie jak metoda podziału i oszacowań, programowanie dynamiczne, algorytmy zachłanne, metody lokalnej poprawy oraz algorytmy programowania linowego.

W rozdziale 3 rozważono podejście przypadku najgorszego do oceny złożoności obliczeniowej zadań i algorytmów optymalizacji dyskretnej. Zdefiniowano niezbędne elementy oceny rozmiaru wielkości danych zadania i nakładu obliczeń, wprowadzono klasy złożoności obliczeniowej.

Podejście przypadku średniego zastało przedstawione w rozdziale 4. Szczególną uwagę poświęcono aparatowi probablistycznemu niezbędnemu w dalszej części monografii oraz prezentacji wybranych wyników analizy przypadku średniego znanych z literatury.

W rozdziałach 5 oraz 6 zaprezentowano wielowymiarowe zadanie załadunku oraz zadanie szeregowania prac z terminami zakończenia. Na przykładzie tych ważnych zadań optymalizacji dyskretnej dokonano dogłębnej analizy zagadnień będących przedmiotem zainteresowania niniejszej monografii, takich jak metody rozwiązywania, analiza złożoności obliczeniowej oraz analiza przypadku średniego. Przedstawione zostały oryginalne wyniki opisujące asymptotyczne zachowanie się rozwiązań optymalnych jako funkcji parametrów zadań w przypadku średnim. Wykazano również, że proste algorytmy przybliżone mogą być asymptotycznie optymalne w sensie analizy przypadku średniego.

Zamiarem autora było używanie możliwie najprostszej notacji matematycznej dla zachowania maksymalnej przejrzystości wywodu. W monografii stosowane są powszechnie przyjęte oznaczenia matematyczne. Dla przykładu:

- $\{a_1, a_2, \ldots, a_n\}$ oznacza zbiór n - elementowy.
- $[x_1, x_2, \ldots, x_m]$ oznacza wektor o m składowych przyjmujących wartości liczbowe.

- $\lim_{x \to a} f(x)$ lub $\lim_{n \to a} g_n$ oznacza granicę funkcji lub ciągu liczbowego, co jednoznacznie wynika z kontekstu.

- $\{X\}$ oznacza jednoznacznie zdefiniowane zdarzenie, na przykład $x < b$, gdzie x jest zmienną, b pewną stałą.

- $|a|$ oznacza wartość bezwzględną liczby a, natomiast $|A|$ oznacza moc (liczbę elementów) zbioru A.

Kolejne oznaczenia są definiowane w tekście monografii w miarę potrzeb. W przypadku możliwości wystąpienia niejednoznaczności w trakcie przeprowadzania wywodów, odpowiednie pojęcia są przytaczane na bieżąco.

Oryginalna terminologia opisująca pojęcia i obiekty rozważane w monografii w przytłaczającej większości pochodzi z języka angielskiego. W monografii wykorzystywana jest polska terminolagia, która zdaniem autora, z jednej strony właściwie oddaje sens oryginału angielskiego, z drugiej zaś strony jest dobrze osadzona w języku polskim. Poza przypadkami oczywistymi, w momencie pierwszego zastosowania nowego, specjalistycznego pojęcia w nawiasach przytoczono jego angielski odpowiednik.

Niniejsza monografia jest efektem wieloletniego zainteresowania autora tematyką analizy przypadku średniego wybranych zadań optymalizacji dyskretnej. Pracę naukową nad tymi zagadnieniami autor prowadził w Instytucie Badań Systemowych Polskiej Akademii Nauk kolejno w Zakładzie Programowania Matematycznego, Pionie Metod Modelowania Matematycznego i Optymalizacji oraz w Pracowni Metod Obliczeniowych Optymalizacji.

Pragnę serdecznie podziękować wszystkim koleżankom i kolegom z IBS PAN za wieloletnią współpracę. Przez cały okres mojej pracy naukowej szczególne znaczenie miała dla mnie współpraca z doc. dr hab. Markiem Liburą, na którego pomoc i cenne rady mogłem zawsze liczyć.

Swojej rodzinie składam wyrazy wdzięczności za cierpliwość, wyrozumiałość i wsparcie podczas całego okresu mojej pracy naukowej. Szczególnie gorąco pragnę podziękować mojej Żonie i Mamie.

Rozdział 3

Złożoność obliczeniowa

3.1 Wprowadzenie

Jak już wspomniano wcześniej, zagadnieniem o podstawowym znaczeniu dla teorii i praktyki rozwiązywania zadań optymalizacji dyskretnej jest złożoność obliczeniowa algorytmów i zadań. Ujmując problem intuicyjnie, jeśli dla wszystkich realizacji danych rozpatrywanego zadania istnieje algorytm dający zawsze rozwiązania optymalne, zaś nakład obliczeń nie jest nadmierny, to takie zadanie możemy uważać za stosunkowo łatwe do rozwiązywania. Natomiast jeżeli algorytm (dokładny czy przybliżony) wymaga dużego nakładu obliczeń, to możliwość wykorzystania go w praktyce obliczeniowej jest bardzo dyskusyjna. Dziedzina nazywana *złożonością obliczeniową* ma, między innymi, za zadanie:

- Określić "efektywność" algorytmów (dokładnych i przybliżonych).
- Dokonać podziału zadań optymalizacji dyskretnej na "łatwiejsze" i "trudniejsze" do rozwiązywania.

W ocenie złożoności obliczeniowej zadań i algorytmów można wyodrębnić dwa zasadnicze podejścia: *analizy przypadku najgorszego* (*worst case analysis*) oraz *analizy przypadku średniego* (*average case analysis*). Podejścia te różnią się zasadniczo swoją filozofią i wykorzystywaną metodologią.

Podejście analizy przypadku najgorszego dla algorytmu polega na wyznaczeniu najbardziej niekorzystnego zestawu danych rozpatrywanego zadania. Uzyskany w ten sposób wynik charakteryzuje algorytm. Jeśli dotyczy on "najlepszego" spośród algorytmów dokładnych dla rozważanego zadania to uzyskujemy w ten sposób ocenę złożoności obliczeniowej zadania. Zaletą tego podejścia jest jego uniwersalność i jednoznaczność. Uzyskane w ten sposób

gwarancje mają charakter absolutny, to znaczy, że ocena działania algorytmów dokładnych dla zadań dotyczy działania rozważanego algorytmu dla wszytkich realizacji danych rozwiązywanego zadania. Wadą tego podejścia jest często uzyskiwana nadmiernie pesymistyczna ocena zarówno algorytmów jak i zadań optymalizacji dyskretnej. W wielu przypadkach uzyskana ocena przypadku najgorszego może być zupełnie niereprezentatywna dla "typowych" realizacji danych zadania.

Celem analizy przypadku średniego jest zbadanie "przeciętnego" zachowania się zarówno zadań jak i algorytmów. Zaletą uzyskiwanych wyników jest określenie przeciętnego, "średniego" zachowania się zadań i algorytmów z odrzuceniem niereprezentatywnych realizacji zadań. Wadą jest, że uzyskuje się wyniki prawdziwe dla zdefiniowanych, losowych klas zadań, co ogranicza ogólność rozważań oraz wymaga zastosowania aparatu probabilistycznego.

W dalszej części tego rozdziału zostanie bardziej szczegółowo omówiona specyfika podejścia przypadku najgorszego. Natomiast w rozdziale 4 przedstawiona zostanie metodologia analizy w sensie przypadku średniego.

3.2 Podejście przypadku najgorszego

W ocenie złożoności obliczeniowej algorytmów ogólnie przyjętą zasadą jest odnoszenie nakładu obliczeń wymaganego przez algorytm do rozmiaru rozwiązywanego zadania. *Rozmiar zadania* jest powszechnie rozumiany jako liczba jego istotnych parametrów. W przypadku zadań programowania liniowego, całkowitoliczbowego, w tym również wielowymiarowych zadań załadunku, patrz (1.1), (1.2), (1.3) oraz (5.1), są to: liczba zmiennych decyzyjnych, na ogół oznaczana jako n, oraz liczba ograniczeń, oznaczana jako m. W przypadku zadań przedstawianych w postaci grafów może to być liczba wierzchołków grafu oraz, ewentualnie, liczba jego łuków lub krawędzi.

Niestety, w wielu przypadkach te charakterystyki nie są wystarczające. W przypadku licznych algorytmów oraz ich komputerowych implementacji ważny jest również zakres zmienności współczynników zadania. Stwierdzenie powyższe ma związek z faktem, że nakład obliczeń niezbędnych dla wykonania podstawowych działań takich jak dodawanie, odejmowanie, porównywanie, mnożenie lub dzielenie dwóch liczb jest zależny od ich wielkości. To znaczy, im większa jest wartość liczb, tym większy nakład obliczeń jest wymagany dla wykonania działania na tych liczbach. Fakt ten znajduje odzwierciedlenie w teoretycznych modelach obliczeń, takich jak *maszyna Turinga* lub *RAM* (*Random Access Machine*), patrz Aho, Hopcroft i Ullman [2], oraz w pracy rzeczywistych komputerów. Jednak w przypadku niezbyt dużych liczb można przyjąć, dla uproszczenia rozważań, że wykonanie podstawowych

działań na tych liczbach wymaga pewnego ustalonego kosztu jednostkowego.

W ogólnym przypadku będziemy mówić, że *rozmiar realizacji danych zadania*, to znaczy wartości wszystkich jego parametrów, jest tożsamy z ilością informacji niezbędną do opisania wszystkich charakterystyk zadania. W przypadku zadania programowania liniowego lub całkowitoliczbowego będą to: liczba zmiennych, liczba ograniczeń, wartości wszystkich współczynników funkcji celu, lewych stron ograniczeń oraz prawych stron ograniczeń. Nieznacznie ograniczając ogólność rozważań można przyjąć założenie, że realizacja danych rozważanego przykładu zadania może być reprezentowana przez poniższy wektor:

$$[\beta_1, \beta_2, \ldots, \beta_v], \text{ gdzie } \beta_i \text{ jest liczbą wymierną, } i = 1, \ldots, v.$$

Aby reprezentację danych zadania przybliżyć zarówno do teoretycznych modeli obliczeń jak również do sposobu działania rzeczywistych komputerów, przyjmiemy założenie, że wszystkie liczby całkowite są reprezentowane w zapisie binarnym. Oznacza to, że liczba całkowita a może zostać zapisana w jednoznaczny sposób w poniższej postaci:

$$a = \sum_{i=0}^{k} \alpha_i \cdot 2^i \text{ gdzie } \alpha_i = 0 \text{ lub } 1, \text{ dla wszystkich } i = 1, \ldots, k. \tag{3.1}$$

Zauważmy, że $k \leq \log_2 a < k + 1$. Znak liczby może być reprezentowany jako wielkość binarna, na przykład $\alpha_{k+1} = 1$ w przypadku liczby dodatniej oraz $\alpha_{k+1} = 0$ w przypadku liczby ujemnej. Aby zapisać liczbę całkowitą a można więc używać wektora binarnego $[\alpha_0, \alpha_1, \ldots, \alpha_k, \alpha_{k+1}]$. Niech c będzie skończoną liczbą wymierną, $c = a, b$, gdzie a jest jej częścią całkowitą oraz b jej częścią ułamkową. Liczba wymierna c może być reprezentowana jako dwie liczby całkowite a oraz b.

Z powyższych rozważań wynika, że każdy wektor $[\beta_1, \beta_2, \ldots, \beta_v]$ liczb wymiernych może zostać zapisany jako wektor binarny

$$[\delta_1, \delta_2, \ldots, \delta_l], \text{ gdzie } \delta_i = 0 \text{ lub } 1, \ i = 1, \ldots, l. \tag{3.2}$$

To oznacza, że każda realizacja danych zadania może być zapisana w jednoznaczny sposób jako wektor binarny o postaci (3.2).

Zadanie optymalizacyjne będziemy teraz rozważać jako $\{p_1, p_2, \ldots, p_i, \ldots\}$ - nieskończony zbiór różnych realizacji jego danych, gdzie każde p_i jest wektorem binarnym $p_i = [\delta_1, \delta_2, \ldots, \delta_{l_i}]$. W przyjętej powyżej konwencji zapisanie realizacji danych zadania p_i wymaga wektora binarnego o rozmiarze $l(p_i) = l_i$. Będziemy mówić, że *rozmiar realizacji danych* zadania p_i wynosi l_i.

Niech A będzie algorytmem wyznaczającym w skończonej liczbie iteracji rozwiązanie optymalne dla każdej z realizacji danych zadania p_i. Niech *nakład obliczeń* wykonywanych przez algorytm A będzie wyrażony jako liczba elementarnych działań wykonywanych przez ten algorytm w celu uzyskania rozwiązania p_i. Przyjmijmy, że nakład obliczeń wykonywanych przez algorytm A jest zadany w postaci poniższej funkcji:

$$g_A(p_i) : \{p_1, p_2, \ldots, p_i, \ldots\} \rightarrow R_+.$$

Zauważmy, że jeżeli dwie realizacje danych zadania $p_{i'}$ oraz $p_{i''}$ mają taki sam rozmiar, $l(p_{i'}) = l(p_{i''})$, to z tego faktu wcale nie wynika, że $g_A(p_{i'}) = g_A(p_{i''})$.

Naszym celem jest uzyskanie oszacowania nakładu obliczeń wymaganego przez algorytm jako funkcję rozmiaru realizacji danych zadania $l_i = l(p_i)$, w taki sposób, aby ocena ta nie zależała od konkretnej postaci p_i. Warunek ten spełnia poniższa funkcja :

$$f_A(n) = \sup\{g_A(p_i) : l(p_i) = n\}. \tag{3.3}$$

Funkcja $f_A(n)$ jest oszacowaniem od góry wymaganego przez algorytm A nakładu obliczeń dla każdej realizacji danych zadania p_i, $i = 1, 2, \ldots$, o rozmiarze n. Ocena ta jest ważna dla wszystkich wartości n, $n = 1, 2, \ldots$.

Definicja 3 *Funkcja $f_A(n)$ zdefiniowana w (3.3) wyznacza złożoność obliczeniową algorytmu A dla rozważanego zadania optymalizacji dyskretnej w sensie analizy przypadku najgorszego.*

Ocena złożoności obliczeniowej algorytmu w sensie analizy przypadku najgorszego ma szereg zalet, poniżej przedstawione zostały najistotniejsze z nich.

- Uzyskana ocena nosi charakter absolutnej gwarancji, a więc jest prawdziwa dla wszystkich realizacji danych zadania.
- Nie jest wymagana żadna dodatkowa wiedza o charakterze zadania i realizacji jego danych, na przykład rozkład prawdopodobieństwa realizacji danych zadania.
- Miara ta wydaje się być najprostszą dla przeprowadzania dalszych analiz, jest ona niezależna od konkretnego modelu maszyny obliczeniowej, takiego jak maszyna Turinga lub RAM, patrz Aho, Hopcroft i Ullman [2], Garey i Johnson [48] oraz Nemhauser i Wolsey [110].

Poza wieloma istotnymi zaletami powyższa ocena złożoności obliczeniowej algorytmu ma jedną kardynalną wadę. Mianowicie wartość $f_A(n)$ uzyskana poprzez (3.3) może być wyznaczona przez realizację danych zadania p_i zbyt pesymistyczną w stosunku do przytłaczającej większości realizacji danych zadania. Oznacza to, że dla większości realizacji danych zadania algorytm może wymagać znacznie mniejszego nakładu obliczeń niż wynika to z oceny danej przez (3.3). W takiej sytuacji może być stosowana analiza przypadku średniego, zaprezentowana w rozdziale 4.

Zamiast uzyskiwania dokładnej postaci funkcji $f_A(n)$ wystarczy dla większości zastosowań rozważać jej asymptotyczne aproksymacje.

Dla dowolnej funkcji $f(n)$, $n = 1, 2, \ldots$, $f(n) \geq 0$, będziemy oznaczać

$$f(n) = O(g(n))$$

jeśli istnieją stałe c' oraz n', $c', n' > 0$, takie, że

$$f(n) \leq c' \cdot g(n) \text{ dla wszystkich } n \geq n'.$$

Dla dowolnej funkcji $f(n)$, $n = 1, 2, \ldots$, $f(n) \geq 0$, będziemy oznaczać

$$f(n) = \Omega(g(n))$$

jeśli istnieją stałe c'' oraz n'', $c'', n'' > 0$, takie, że

$$c'' \cdot g(n) \leq f(n) \text{ dla wszystkich } n \geq n''.$$

Dla dowolnej funkcji $f(n)$, $n = 1, 2, \ldots$, $f(n) \geq 0$, będziemy oznaczać

$$f(n) = \Theta(g(n))$$

jeśli

$$f(n) = O(g(n)) \text{ oraz } f(n) = \Omega(g(n)).$$

Zgodnie z powyższą notacją dla funkcji wielomianowej $f(n) = \sum_{i=0}^{k} a_i \cdot n^i$, $a_i > 0$, mamy

$$f(n) = O(n^k),\ f(n) = \Omega(n^k) \text{ oraz } f(n) = \Theta(n^k).$$

Powyższy przykład pokazuje, że oceny $O(\cdot)$, $\Omega(\cdot)$ oraz $\Theta(\cdot)$ mają charakter asymptotycznej aproksymacji zachowania funkcji $f(n)$. Przy $n \to \infty$ pomijane są wszystkie potęgi mniejsze niż k oraz stałe mnożniki.

Algorytmy o gwarantowanej dokładności i złożoności obliczeniowej

Pojęcie złożoności obliczeniowej w połączeniu z oszacowaniem dokładności działania algorytmów w sensie miar wprowadzonych w podrozdziale 2.1, patrz (2.1) oraz (2.3), pozwala wyodrębnić klasę algorytmów przybliżonych dla zadań optymalizacji dyskretnej posiadających gwarancje dotyczące zarówno dokładności ich działania jak również złożoności obliczeniowej.

Definicja 4 *Algorytm przybliżony A nazywamy wielomianowym schematem aproksymującym (polynomial time approximation scheme) jeżeli dla dowolnego $0 < \epsilon < 1$ uzyskuje on rozwiązanie przybliżone zadania takie, że spełnione jest (2.3) oraz $f_A(n)$ jest ograniczona przez funkcję wielomianową od n, $f_A(n) = O(n^k)$, gdzie $k \geq 1$ jest stałą.*

Definicja 5 *Algorytm przybliżony A nazywamy w pełni wielomianowym schematem aproksymującym (fully polynomial time approximation scheme) jeżeli dla dowolnego $0 < \epsilon < 1$ uzyskuje on rozwiązanie przybliżone zadania takie, że spełnione jest (2.3) oraz $f_A(n)$ jest ograniczona przez funkcję wielomianową od n oraz $1/(1-\epsilon)$.*

3.3 Algorytmy wielomianowe i klasa $\mathcal{P}$

Funkcje wielomianowe rosną stosunkowo powoli wraz ze wzrostem n. W związku z tym szczególną uwagę poświęcono algorytmom o wielomianowej złożoności obliczeniowej i zadaniom optymalizacji dyskretnej, dla których one istnieją.

Definicja 6 *Algorytm A dla zadania optymalizacyjnego będziemy nazywać algorytmem o złożoności wielomianowej (polynomial time algorithm), jeśli*

$$f_A(n) = O(n^k) \text{ gdzie } k, k \geq 1, \text{ jest stałą}$$

dla wszystkich realizacji danych zadania $\{p_1, p_2, \ldots, p_i, \ldots\}$.

Definicja 7 *Klasa zadań optymalizacji dyskretnej, dla których istnieją dokładne algorytmy o złożoności wielomianowej, nazywana jest klasą $\mathcal{P}$.*

Zadania z klasy $\mathcal{P}$ są uważane za stosunkowo proste do rozwiązywania z powodu niezbyt dużego oczekiwanego nakładu obliczeniowego. W związku z potrzebą efektywnego rozwiązywania zadań optymalizacji dyskretnej zagadnieniem o zasadniczym znaczeniu jest zbadanie różnic pomiędzy zadaniami należącymi do klasy $\mathcal{P}$ oraz zadaniami, dla których nie jest znany żaden dokładny algorytm wielomianowy.

Definicja 8 *Funkcję $f(n)$, $n = 1, 2, \ldots$, $f(n) \geq 0$, będziemy nazywać funkcją wykładniczą, jeżeli istnieje stała $\alpha > 1$ taka, że*

$$f(n) = \alpha^n$$

Definicja 9 *Algorytm A dla zadania optymalizacyjnego będziemy nazywać algorytmem o złożoności wykładniczej (exponential time algorithm), jeśli*

$$f_A(n) = \Omega(\alpha^n) \text{ oraz } f_A(n) = O(\beta^n) \text{ gdzie } \alpha, \beta > 1, \text{ są stałymi.}$$

Przykładem procesu obliczeniowego o złożoności obliczeniowej wykładniczej jest przegląd wszystkich 2^n wektorów binarnych o postaci $[x_1, \ldots, x_n]$, gdzie $x_i = 0$ lub 1. Funkcja 2^n rośnie bardzo szybko, nie istnieje taka stała k, że

$$2^n \leq n^k \text{ dla wszystkich } n,$$

a więc nie istnieje funkcja wielomianowa ograniczająca od góry wzrost funkcji wykładniczej. Aby zilustrować szybkość wzrostu funkcji wykładniczej rozważmy następujący przykład. Jeśli każda z wykonywanych operacji trwa jedną mikrosekundę i wykonywane jest 2^n operacji, to przy $n = 60$ czas obliczeń przekroczy **300 wieków** (30.000 lat), podczas gdy dla funkcji n^5 wyniesie on poniżej 15 minut.

Większość znanych algorytmów ma złożoność obliczeniową wielomianową albo wykładniczą. Warto jednak mieć świadomość, że istnieją również:

- funkcje rosnące szybciej niż funkcje wielomianowe ale wolniej niż wykładnicze, na przykład $f(n) = n^{\log n}$ oraz
- funkcje rosnące szybciej niż funkcje wykładnicze, na przykład $f(n) = n^{n^n}$.

Zauważmy, że algorytm sympleks dla zadania programowania liniowego, patrz Klee i Minty[86] oraz podrozdział 2.5, ma w przypadku najgorszym wykładniczą złożoność obliczeniową, co nie umniejsza jego wysokiej przydatności praktycznej. Algorytm elipsoidalny zaproponowany w pracy Chacziana [20] ma wielomianową złożoność obliczeniową. Ten wynik pozwolił zakwalifikować zadanie programowania liniowego do klasy $\mathcal{P}$. Natomiast przydatność praktyczna algorytmu Chacziana okazała się bardzo wątpliwa, gdyż został w nim zastosowany bardzo szczególny model obliczeniowy.

3.4 Algorytm niedeterministyczny i klasa $\mathcal{NP}$

Jak już wspomniano powyżej, funkcje wielomianowe rosną stosunkowo powoli, szczczególnie w porównaniu do funkcji wykładniczych. Dlatego zagadnieniem o szczególnym znaczeniu jest stwierdzenie, czy dla danego zadania optymalizacji dyskretnej może istnieć algorytm uzyskujący zawsze rozwiązania optymalne i mający wielomianową złożoność obliczeniową. Fakt przynależności zadania do klasy $\mathcal{P}$ decyduje o względnej łatwości jego rozwiązywania. Niestety, w odniesieniu do wielu ważnych zadań optymalizacji dyskretnej, takich na przykład jak programowanie całkowitoliczbowe, zadanie komiwojażera i wiele innych, nie są znane dokładne algorytmy wielomianowe. Celem zaprezentowanego poniżej modelu teoretycznego było przeanalizowanie możliwości istnienia dla takich zadań algorytmów wielomianowych. Chociaż model ten nie rozstrzyga jednoznacznie faktu nieistnienia dla szczególnie trudnych zadań optymalizacji dyskretnej algorytmów wielomianowych, to jednak zasadniczym wnioskiem, jaki można wyciągnąć na jego podstawie, jest stwierdzenie, iż z bardzo wysokim stopniem wiarygodności istnieją zadania optymalizacji dyskretnej nie należące do "łatwiejszej" klasy $\mathcal{P}$.

Definicja 10 *Problem rozstrzygalności (feasibility problem) dla zadania X stanowi para (D, V) gdzie $V \subseteq D$ oraz elementy zbioru D są wektorami binarnymi. Zbiór D będziemy nazywać zbiorem realizacji rozwiązań dla zadania X, natomiast V będziemy nazywać zbiorem dopuszczalnych realizacji rozwiązań dla zadania X. Dla danego $d \in D$ chcemy znać odpowiedź na pytanie, czy $d \in V$. Dla każdego $d \in D$ odpowiedź brzmi tak lub nie.*

W odniesieniu do zadań optymalizacji dyskretnej odpowiedni problem rozstrzygalności jest definiowany w taki sposób, aby zbiór V zawierał jednoznaczny opis zbioru rozwiązań dopuszczalnych rozważanego zadania optymalizacji dyskretnej. Dla jednowymiarowego zadania załadunku (1.6) zbiór D składa się ze wszystkich możliwych 2^n wektorów binarnych o postaci $[x_1, \ldots, x_n]$ gdzie $x_i = 0$ lub 1, natomiast zbiór V zawiera wektory binarne spełniające $\sum_{i=1}^{n} a_i \cdot x_i \leq b(n)$, a więc zbiór V zawiera zbiór rozwiązań dopuszczalnych dla realizacji danych zadania.

Zbiory D oraz V mogą być konstruowane w taki sposób, aby dodatkowo zawierały zależności opisujące wartość kryterium optymalizacyjnego (funkcji celu) dla każdego rozwiązania dopuszczalnego realizacji danych zadania. Rozważania powyższe mogą być zilustrowane na przykładzie jednowymiarowego zadania załadunku (1.6). W konstrukcji zbioru V do opisu zbioru rozwiązań dopuszczalnych realizacji danych zadania dodajemy dodatkowe ograniczenie o postaci:

$$z_{OPT}(n) \geq \omega.$$

W tym przypadku dla każdej konkretnej wartości parametru ω istnienie $d \in V$ oznacza, że istnieje rozwiązanie dopuszczalne jednowymiarowego zadania załadunku (1.6) o wartości funkcji celu nie mniejszej niż ω. Przyjmując założenie o całkowitoliczbowości i nieujemności współczynników funkcji celu zadania c_i, rozwiązanie optymalne zadania załadunku może zostać wyznaczone po sprawdzeniu co najwyżej $\sum_{i=1}^{n} c_i + 1$ różnych wartości ω, $\omega = 0, 1, 2, \cdots, \sum_{i=1}^{n} c_i$.

Klasa $\mathcal{P}$ jest w tym ujęciu rozumiana jako klasa problemów rozstrzygalności rozwiązywalnych przez algorytmy o wielomianowej złożoności obliczeniowej.

Aby wprowadzić pojęcie algorytmu niedeterministycznego potrzebne jest wprowadzenie kilku dodatkowych pojęć. Jednym z nich jest *świadectwo rozstrzygalności* (*certificate of feasibility*).

Definicja 11 *Niech Q_d oznacza wektor binarny, za pomocą którego dla każdego $d \in D$, jeżeli $d \in V$, uzyskiwana jest odpowiedź pozytywna. Q_d nazywamy świadectwem rozstrzygalności dla (D, V).*

Powstaje naturalne pytanie, jak praktycznie możemy zrealizować świadectwo rozstrzygalności Q_d. W tym celu wyobraźmy sobie algorytm, który generuje dużą liczbę zapytań w celu ewentualnego odgadnięcia Q_d. Ta idea prowadzi wprost do pojęcia *algorytmu niedeterministycznego* (*nondeterministic algorithm*). Należy mieć na uwadze, że algorytmy tego typu są niekonstruktywne, to znaczy, że nie mogą być one zaimplementowane w praktyce obliczeniowej. Działanie algorytmu niedeterministycznego składa się z dwóch faz. Dane wejściowe dla algorytmu niedeterministycznego stanowi $d \in D$.

1. W pierwszej, *zgadującej* fazie (*guessing stage*) swojej pracy algorytm zgaduje (inaczej - korzysta z wyroczni) ciąg binarny Q, który jest przekazywany do drugiej, *weryfikującej* fazy (*checking stage*) jego pracy.

2. W fazie weryfikującej algorytm pracuje z parą (d, Q) i może stwierdzić, że $d \in V$. Wymagane są dwie właściwości:

   - Jeżeli $d \in V$, to istnieje świadectwo rozstrzygalności Q_d takie, że kiedy para (d, Q_d) zostanie przekazana do fazy weryfikującej, to algorytm da odpowiedź, że $d \in V$.
   - W przypadku $d \notin V$ algorytm nie uzyskuje żadnego wyniku, jego działanie jest niezdefiniowane, może on nawet nie zakończyć pracy. Tak więc zakończenie pracy algorytmu oznacza, że $d \in V$.

Złożoność obliczeniowa algorytmu niedeterministycznego jest mierzona tylko w fazie weryfikującej i tylko w przypadku istnienia świadectwa rozstrzygalności $d \in V$.

Definicja 12 *Niedeterministyczny algorytm A ma wielomianową złożoność obliczeniową, jeżeli dla każdego $d \in V$ nakład wykonywanych przez niego obliczeń jest ograniczony przez funkcję będącą wielomianem od rozmiaru wektora binarnego reprezentującego d dla pewnego świadectwa rozstrzygalności Q_d, gdzie Q_d pozwala stwierdzić, że $d \in V$.*

Sens powyższej definicji jest taki, że jeżeli $d \in V$, to algorytm niedeterministyczny stwierdza to szybko, to znaczy w wielomianowym nakładzie obliczeń. Innymi słowami można powiedzieć, że istnieje w tym przypadku *krótki dowód rozstrzygalności* (*short proof of feasibility*). Poniżej zostanie zdefiniowana klasa zadań $\mathcal{NP}$. Odgrywa ona zasadniczą rolę w teorii złożoności obliczeniowej.

Definicja 13 *Klasa problemów rozstrzygalności, gdzie dla każdego $d \in V$ istnieje wielomianowy, niedeterministyczny algorytm rozstrzygający fakt, że $d \in V$, nazywana jest klasą $\mathcal{NP}$.*

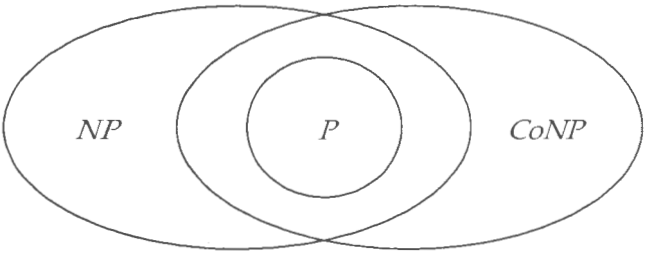
Skrót $\mathcal{NP}$ oznaczał pierwotnie *niedeterministyczny wielomianowy (nondeterministic polynomial)*. Obecnie coraz więcej autorów odczytuje go jako *niewielomianowy* (*non polynomial*). Zauważmy, że dla każdego $d \in V$ istnieje wektor binarny Q_d, którego rozmiar $l(Q_d)$ jest wielomianem od rozmiaru d. Niech $L = l(Q_d) = O(l^p(d))$. Wynika z tego, że dla każdego konkretnego $d \in D$ w fazie zgadującej wystarczy rozważenie wektorów binarnych o rozmiarach nie przekraczających L. Dlatego wystarczy rozważyć co najwyżej 2^{L+1} wektorów, aby roztrzygnąć, że $d \in V$. Wszystkie pozostałe elementy wielomianowego, niedeterministycznego algorytmu rozstrzygającego mają wielomianową złożoność obliczeniową od rozmiaru danych wejściowych. Oznacza to, że działanie wielomianowego niedeterministycznego algorytmu rozstrzygającego może być modelowane przez algorytm deterministyczny o co najwyżej wykładniczej złożoności obliczeniowej od rozmiaru danych wejściowych zadania.

Dla każdego problemu rozstrzygalności $X = (D, V)$ istnieje związany z nim problem rozstrzygalności $\bar{X} = (D, \bar{V})$, gdzie $\bar{V} = D \backslash V$. $\bar{X}$ zwany jest *dopełnieniem* X. Istnieje również klasa zadań będąca dopełnieniem klasy $\mathcal{NP}$, w skład której wchodzą zadania nie należące do klasy $\mathcal{NP}$.

Definicja 14 *Klasa $Co\mathcal{NP}$ jest to klasa problemów rozstrzygalności X takich, że ich dopełnienia należą do klasy $\mathcal{NP}$, co można zapisać w poniższy sposób:*

$$Co\mathcal{NP} = \{X : \bar{X} \in \mathcal{NP}\}.$$

Zauważmy, że jeżeli problem rozstrzygalności X należy do klasy $\mathcal{P}$, to również jego dopełnienie $\bar{X}$ należy do klasy $\mathcal{P}$. Z drugiej strony dla dopełnień wielu problemów rozstrzygalności, na przykład dla problemu rozstrzygalności związanego z zadaniem programowania binarnego, nie wiadomo, czy ich dopełnienia należą do klasy $\mathcal{NP}$. Poniższy rysunek 3.1 przedstawia wzajemne, możliwe relacje pomiędzy klasami $\mathcal{P}$, $\mathcal{NP}$ oraz $\mathcal{CoNP}$.



Rysunek 3.1: Relacje pomiędzy klasami $\mathcal{NP}$, $\mathcal{CoNP}$ oraz $\mathcal{P}$

Do klasy $\mathcal{NP}$ należy wiele ważnych zadań optymalizacji dyskretnej, takich jak zadania programowania mieszanego, całkowitoliczbowego i liniowego, w tym zadania załadunku, patrz podrozdział 1.2, zadania rozbicia zbioru, zadanie pakowania zasobników, patrz podrozdział 1.3, liczne zadania teorii grafów, na przykład zadanie komiwojażera, zadanie wyznaczenia najkrótszego drzewa rozpinającego w grafie ważonym, patrz podrozdział 1.4, wiele zadań harmonogramowania, patrz podrozdział 1.5.

Dla wielu zadań optymalizacji dyskretnej istnieją algorytmy wielomianowe, na przykład dla zadań programowania liniowego oraz dla zadania wyznaczenia najkrótszego drzewa rozpinającego w grafie ważonym. Należą one do klasy zadań $\mathcal{P}$.

Dla wielu istotnych zadań optymalizacji dyskretnej, na przykład zadań programowania całkowitoliczbowego, komiwojażera oraz wielu zadań teorii harmonogramowania nie są znane algorytmy wielomianowe. Co więcej, doświadczenie wielu badaczy wskazuje, że ich istnienie jest bardzo problematyczne. Oznacza to, że w klasie $\mathcal{NP}$ mogą istnieć zadania nie należące do klasy $\mathcal{P}$.

Jest faktem oczywistym, że

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Pytaniem o zasadniczym znaczeniu dla oceny złożoności obliczeniowej licznych ważnych zadań optymalizacji jest pytanie, czy

$$\mathcal{P} = \mathcal{NP}. \tag{3.4}$$

Fakt ten jest wciąż nie rozstrzygnięty, ale powszechnie uważa się, że w klasie $\mathcal{NP}$ istnieją zadania nie należące do klasy $\mathcal{P}$ a więc, że

$$\mathcal{P} \neq \mathcal{NP}. \tag{3.5}$$

Jeżeli prawdziwa jest relacja (3.5) to w klasie $\mathcal{NP}$ mogą istnieć klasy zadań trudniejszych niż klasa $\mathcal{P}$, dla których nie istnieją algorytmy wielomianowe. Poniżej idea ta zostanie rozwinięta.

Jeżeli mamy rozwiązać nowe zadanie optymalizacyjne, to dość powszechnie stosowanym podejściem jest próba sprowadzenia rozważanego zadania do innego, znanego zadania i zastosowanie wykorzystywanych dla niego algorytmów do rozwiązania nowego zadania. Podobna idea jest stosowana do problemów rozstrzygalności z klasy $\mathcal{NP}$. Załóżmy, że mamy dwa problemy rozstrzygalności $X_1 = (D_1, V_1)$ i $X_2 = (D_2, V_2)$ oraz, że istnieje funkcja $g : D_1 \to D_2$, taka, że dla każdego $d \in D_1$, $g(d) \in V_2$ wtedy i tylko wtedy, kiedy $d \in V_1$. Jeżeli wartość $g(d)$ dla wszystkich $d \in D_1$ jest obliczana w czasie ograniczonym przez wielomian od rozmiaru d, to wtedy X_1 jest *wielomianowo transformowalne* do X_2. Jeśli X_1 jest wielomianowo transformowalne do X_2 oraz X_2 jest rozstrzygalny w wielomianowym czasie, to X_1 jest również rozstrzygalny w wielomianowym czasie. Natomiast z faktu wielomianowej rozstrzygalności X_1 nie wynika wielomianowa rozstrzygalność X_2. Będziemy mówić, że X_1 jest *wielomianowo sprowadzalne* do X_2, jeśli dla rozstrzygnięcia X_1 istnieje algorytm wykorzystujący algorytm dla rozstrzygnięcia X_2 jako procedurę. Algorytm ten działa w czasie wielomianowym zależnym od rozmiaru danych wejściowych. Wspomniana procedura rozstrzygająca jest wywoływana w jednostkowym koszcie nakładu obliczeń. Wielomianową sprowadzalność X_1 do X_2 będziemy oznaczać

$$X_1 \propto X_2.$$

Pojęcia wielomianowej transformowalności i wielomianowej sprowadzalności są sobie bliskie. Wielomianowa transformowalność jest tożsama z wielomianową sprowadzalnością z jedorazowym wywołaniem wzmiankowanej procedury. Konsekwencje dla X_1 wielomianowo sprowadzalnego do X_2 są podobne jak powyżej opisane dla wielomianowej transformowalności. Mianowicie:

- jeżeli $X_1 \propto X_2$ oraz $X_2 \in \mathcal{P}$, to wtedy również $X_1 \in \mathcal{P}$.

Poniżej przedstawimy definicję problemów istotnie trudniejszych od klasy $\mathcal{P}$, przy założeniu prawdziwości (3.5).

Definicja 15 *Problem rozstrzygalności $X \in \mathcal{NP}$ będziemy zazywać $\mathcal{NP}$-zupełnym ($\mathcal{NP}$-complete), jeżeli wszystkie problemy z klasy $\mathcal{NP}$ są wielomianowo sprowadzalne do problemu X. Klasę problemów $\mathcal{NP}$-zupełnych będziemy oznaczać $\mathcal{NPC}$.*

Zauważmy, że:

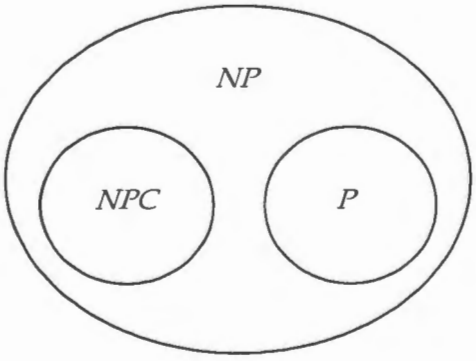
jeżeli $X_1 \propto X_2$ oraz $X_1 \in \mathcal{NPC}$, to wtedy również $X_2 \in \mathcal{NPC}$.

Oznacza to, że każdy problem rozstrzygalności, do którego jest wielomianowo sprowadzalny problem $\mathcal{NP}$-zupełny, jest również problemem $\mathcal{NP}$-zupełnym.

Przyjmując, że klasa $\mathcal{NPC}$ zawiera najtrudniejsze problemy z klasy $\mathcal{NP}$ powstaje pytanie o wzajemne relacje klas $\mathcal{NPC}$ oraz $\mathcal{P}$. Zachodzi następująca relacja:

$\mathcal{P} = \mathcal{NP}$ wtedy i tylko wtedy, jeżeli istnieje $X : X \in \mathcal{NPC}$ oraz $X \in \mathcal{P}$.

Ogólnie, zakładając prawdziwość (3.5), realacje pomiędzy klasami $\mathcal{NP}$, $\mathcal{NPC}$ oraz $\mathcal{P}$ są przedstawione na rysunku 3.2.



Rysunek 3.2: Relacje pomiędzy klasami $\mathcal{NP}$, $\mathcal{NPC}$ oraz $\mathcal{P}$

Zbiór problemów $\mathcal{NP}$-zupełnych nie jest pusty. Pierwszy problem $\mathcal{NP}$-zupełny przedstawił Cook w roku 1971 w pracy [26], następne Karp w roku 1972 w pracy [79]. Książka Garey'a i Johnsona [48] jest bogatym kompendium wiedzy o problemach $\mathcal{NP}$-zupełnych. Nowo poznane problemy $\mathcal{NP}$-zupełne są na bieżąco prezentowane, na przykład w pracach D.S. Johnson [71], [72] oraz [74].

Jak już wspomniano na stronie 72, dla zadań optymalizacji dyskretnej istnieją odpowiadające im problemy rozstrzygalności.

Definicja 16 *Jeżeli problem rozstrzygalności odpowiadający zadaniu optymalizacji dyskretnej jest $\mathcal{NP}$-zupełny, to zadanie optymalizacyjne będziemy nazywać $\mathcal{NP}$-trudnym ($\mathcal{NP}$-hard).*

Określenia $\mathcal{NP}$-zupełny oraz $\mathcal{NP}$-trudny są często stosowane wymiennie w odniesieniu do zadań optymalizacji dyskretnej. Należy mieć na uwadze, iż użycie jednego z tych terminów oznacza, że problem rozstrzygalności odpowiadający zadaniu optymalizacj dyskretnej jest $\mathcal{NP}$-zupełny.

$\mathcal{NP}$-trudnymi zadaniami optymalizacji dyskretnej są, między innymi, wymienione powyżej zadania programowania mieszanego, całkowitoliczbowego w tym zadania załadunku, zadania rozbicia zbioru, zadanie pakowania zasobników, zadanie komiwojażera i wiele innych.

Jak wspomniano w podrozdziale 3.2, dane zadań są zapisywane w postaci binarnej, patrz (3.1). Oznacza to, że dla zapisu liczby całkowitej a potrzebujemy ciągu binarnego o długości $\lfloor \log_2 a \rfloor + 1$. Innym możliwym sposobem zapisu liczby całkowitej a jest *kodowanie unarne*, a jest wtedy reprezentowana przez ciąg jedynek o długości $|a| + 1$ (to znaczy $|a|$ jedynek plus dodatkowy symbol 0 lub 1 na oznaczenie znaku liczby).

Definicja 17 *Algorytm wymagający wielomianowego nakładu obliczeniowego przy zastosowaniu unarnego kodowania danych zadania nazywamy algorytmem pseudowielomianowym (pseudopolynomial algorithm). Klasę zadań, dla których istnieją optymalne algorytmy pseudowielomianowe będziemy nazywać Pseudo-$\mathcal{P}$.*

Zauważmy, że w pewnych przypadkach algorytm pseudowielomianowy może stać się algorytmem wielomianowym, w innych zaś wykładniczym. W poniższej definicji pojęcia algorytmu psudowielomianowego użyto do zdefiniowania szczególnie trudnej podklasy problemów $\mathcal{NP}$-zupełnych.

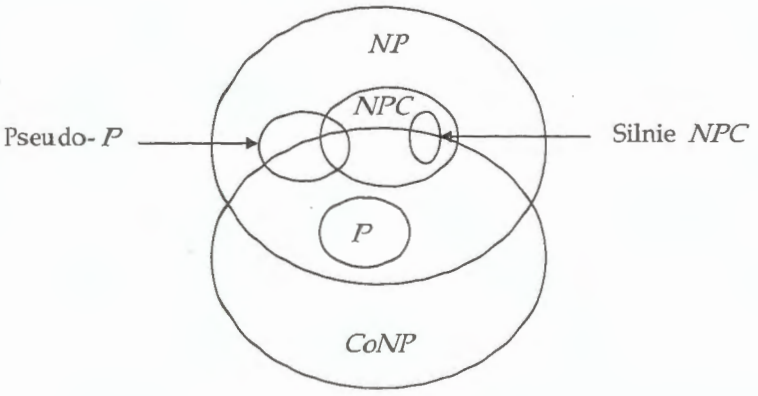
Definicja 18 *Problemy $\mathcal{NP}$-zupełne, dla których istnienie dokładnych algorytmów pseudowielomianowych oznacza $\mathcal{NP} = \mathcal{P}$, będziemy nazywali silnie $\mathcal{NP}$-zupełnymi (strongly $\mathcal{NP}$-complete) i oznaczać Silnie $\mathcal{NPC}$. Odpowiadające im zadania optymalizyjne będziemy nazywać silnie $\mathcal{NP}$-trudnymi.*

Wzajemne relacje pomiędzy różnymi klasami złożoności problemów rozstrzygalności, a więc również i zadań optymalizacyjnych, są przedstawione na rysunku 3.3.

Ścisłe, w sensie równości czy zawierania, relacje pomiędzy klasami $\mathcal{NP}$, $\mathcal{NPC}$, $\mathcal{CoNP}$, $\mathcal{P}$, Silnie-$\mathcal{NPC}$ oraz Pseudo-$\mathcal{P}$ nie są znane. Jeżeli zachodzi relacja (3.4), to wtedy

$$\mathcal{NP} = \mathcal{NPC} = \mathcal{CoNP} = \text{Silnie-}\mathcal{NPC} = \mathcal{P} = \text{Pseudo-}\mathcal{P}.$$

Prawdziwość jednej z dwóch wykluczających się relacji (3.5) lub (3.4) pomimo ponad 25 lat wysiłków wielu badaczy jest wciąż otwartym problemem



Rysunek 3.3: Relacje klas $\mathcal{NP}$, $\mathcal{NPC}$, $\mathcal{CONP}$, $\mathcal{P}$, Silnie $\mathcal{NPC}$ i Pseudo-$\mathcal{P}$

poznawczym o kluczowym znaczeniu dla teorii złożoności obliczeniowej, optymalizacji dyskretnej, badań operacyjnych, informatyki, matematyki i wielu pokrewnych dziedzin.

Powszechnie oczekiwany jest dowód, że prawdziwa jest relacja (3.5). Prawdziwość (3.5) oznacza, że istnieją zadania optymalizacji dyskretnej o złożoności obliczeniowej wyższej niż wielomianowa. Oczekiwanie to wynika z dotychczasowej wiedzy o zadaniach optymalizacji dyskretnej, na przykład o zadaniach komiwojażera lub programowania całkowitoliczbowego.

Jeżeli prawdziwa jest relacja (3.5), to nie wyklucza ona możliwości efektywnego rozwiązywania oddzielnych przypadków $\mathcal{NP}$-trudnych zadań nawet o dużych rozmiarach. Natomiast z ewentualnej prawdziwości relacji (3.4) nie będzie automatycznie wynikała łatwość rozwiązywania wszystkich zadań optymalizacji dyskretnej w praktyce obliczeniowej. Z powyższych rozważań wypływa wniosek, że prawdziwość jednej z relacji (3.5) lub (3.4) nie wyklucza istnienia pewnych przypadków szczególnych odbiegających od normy, którą wyznacza przynależność zadania do określonej klasy złożoności obliczeniowej.

Uwagi powyższe w żaden sposób nie umniejszają olbrzymiej wartości poznawczej klasyfikacji zadań optymalizacji dyskretnej, uzyskanej w oparciu o teorię złożoności obliczeniowej. Ponadto, zjawiska te mogą być dokładniej zrozumiane i wyjaśnione przy zastosowaniu analizy przypadku średniego rozważanej w następnych rozdziałach monografii.

3.5 Podsumowanie

Celem tego rozdziału było przedstawienie podstawowych pojęć złożoności obliczniowej zadań i algorytmów optymalizacji dyskretnej zgodnie z podejściem przypadku najgorszego. Zostały formalnie zdefinowane między innymi takie pojęcia jak:

- Rozmiar zadania optymalizacji dyskretnej.
- Nakład obliczeń jako funkcja rozmiaru zadania. Dokonano rozróżnienia na algorytmy o "niskim" (wielomianowym) i "wysokim" (wykładniczym) nakładzie obliczeń.
- Złożoność obliczeniowa algorytmów optymalizacji dyskretnej.
- Klasa zadań $\mathcal{P}$ o wielomianowej złożoności obliczeniowej.
- Klasa zadań $\mathcal{NP}$ zawierająca zadania optymalizacji dyskretnej o złożoności obliczeniowej wyższej niż wielomianowa, o ile prawdziwa jest relacja $\mathcal{P} \neq \mathcal{NP}$.
- W ramach klasy $\mathcal{NP}$ wyodrębniono podklasy zadań szczególnie trudnych, takich jak zadania $\mathcal{NP}$-trudne oraz silnie $\mathcal{NP}$-trudne.

Zamiarem autora było zdefiniowanie podstawowych pojęć złożoności obliczeniowej zadań i algorytmów optymalizacji dyskretnej w sensie analizy przypadku najgorszego. Teoria ta pozwala umownie podzielić zadania optymalizacji dyskretnej na łatwe, trudne i bardzo trudne. W rzeczywistości analiza złożoności obliczeniowej może być rozszerzona na wiele dodatkowych klas złożoności obliczeniowej, na przykład w oparciu o pojęcie *złożoności przestrzennej* (space complexity), gdzie oprócz niezbędnego nakładu obliczeń analizowane jest wykorzystanie pamięci (rozmiaru zapisu danych) przez algorytmy.

W rozdziałach 4, 5 oraz 6 zaprezentowano wyniki pokazujące, że pomimo przynależności zadań do klasy problemów $\mathcal{NP}$-trudnych lub nawet silnie $\mathcal{NP}$-trudnych, mogą dla nich istnieć algorytmy asymptotycznie optymalne w sensie analizy przypadku średniego.

Uwagi końcowe

W monografii rozważono podejście przypadku średniego do analizy zadań oraz algorytmów optymalizacji dyskretnej. Celem monografii było wykazanie, na przykładzie binarnego wielowymiarowego zadania załadunku, zadania szeregowania prac z terminami zakończenia oraz wyników znanych z literatury, że podejście przypadku średniego jest bardzo użytecznym narzędziem analizy zadań i algorytmów optymalizacji dyskretnej.

W monografii dokonano przeglądu dziedziny optymalizacji dyskretnej z zaprezentowaniem najbardziej charakterystycznych zadań, takich jak: programowanie całkowitoliczbwe i liniowe, programowanie binarne, zadania pokrycia, pakowania i rozbicia zbiorów, wybrane zadania teorii grafów oraz zadania harmonogramowania.

Następnie zaprezentowano popularne i powszechnie stosowane techniki rozwiązywania zadań optymalizacji dyskretnej, takie jak: metoda pełnego przeglądu, metoda podziału i oszacowań, programowanie dynamiczne, algorytmy zachlanne, metody programowania liniowego i inne.

Zdefinowane zostały sposoby oceny dokładności pracy algorytmów optymalizacji dyskretnej. Do oceny złożoności obliczeniowej zadań i algorytmów optymalizacji dyskretnej wprowadzono metodologię przypadku najgorszego. Dokonano podziału zadań optymalizacji na umowne kategorie zadań łatwych (należących do klasy $\mathcal{P}$), trudnych (należących do klasy zadań $\mathcal{NP}$-trudnych) oraz szczególnie trudnych (zadań silnie $\mathcal{NP}$-trudnych). Z pewnym uproszczeniem można powiedzieć, że o łatwości rozwiązywania wybranych zadań optymalizacji dyskretnej decyduje istnienie dla nich algorytmów dokładnych o wielomianowej złożoności obliczeniowej.

Jako dopełnienie oraz poszerzenie możliwości poznawczych podejścia przypadku najgorszego zaprezentowano podejście przypadku średniego. Zdefiniowano niezbędne podstawy teoretyczne analizy przypadku średniego oraz dokonano prezentacji wybranych wyników znanych z literatury.

Ogólne rozważania dotyczące zadań optymalizacji dyskretnej, metod ich rozwiązywania, podejścia analizy przypadku najgorszego i średniego do oceny zadań i algorytmów optymalizacji dyskretnej szczegółowo omówiono na przy-

kładzie wielowymiarowego binarnego zadania załadunku, z odrębnym rozpatrzeniem przypadku jednowymiarowego oraz zadania szeregowania prac z terminami zakończenia.

Dla rozważanych modeli losowych zadań, które uzyskano przyjmując założenie, że współczynniki funkcji celu i lewych stron ograniczeń są realizacjami zmiennych losowych o rozkładzie równomiernym w przedziale $(0, 1]$, uzyskano szereg ciekawych wyników analizy przypadku średniego. Najważniejszym z nich było wykazanie, że wartości rozwiązań optymalnych dla całych losowych klas zadań dążą do swoich wartości oczekiwanych - deterministycznych funkcji: rozmiaru zadania n (liczby zmiennych decyzyjnych), liczby ograniczeń m (w przypadku binarnego wielowymiarowego zadania załadunku), oraz wartości prawych stron ograniczeń. Z przeprowadzonych rozważań wynika istotny wpływ wartości i wzajemnych uwarunkowań wektorów prawych stron ograniczeń na asymptotyczny wzrost wartości rozwiązań optymalnych jako funkcji rozmiaru zadania n.

W przypadku binarnego wielowymiarowego zadania załadunku można zauważyć, że liczba ograniczeń m ma bardzo duży wpływ na asymptotyczny wzrost wartości rozwiązań optymalnych jako funkcji rozmiaru zadania n, szczególnie w przypadku funkcyjnej zależności wartości prawych stron ograniczeń od m oraz małych wartości prawych stron ograniczeń $b_j(n)$, $j = 1, \ldots, m$. Dla dużych wartości $b_j(n)$, $j = 1, \ldots, m$, zależność od m ulega znacznemu osłabieniu, a przy $b_j(n) \approx n/2$ praktycznie zanika.

Powszechnie stosowaną w praktyce obliczeniowej metodą służącą do oceny algorytmów przybliżonych jest testowanie ich na zadaniach generowanych losowo. Uzyskane wyniki są następnie poddawane analizie statystycznej. Łatwo jest zauważyć, że powszechnie stosowane generatory zadań losowych są praktycznie tożsame z rozważanymi w monografii losowymi modelami zadań. Wyniki analizy przypadku średniego są więc w wielu przypadkach potwierdzeniem i teoretycznym uzasadnieniem wyników eksperymentalnych.

Co więcej, w uzasadnionych przypadkach wyniki analizy przypadku średniego mogą wyeliminować konieczność przeprowadzania eksperymentu obliczeniowego testującego algorytmy dla zadań optymalizacji dyskretnej. W takim przypadku w oczywisty sposób jest oszczędzany czas badaczy oraz zmniejszane wykorzystanie zasobów komputerowych.

W monografii wykazano, że bardzo proste algorytmy heurystyczne, o liniowej złożoności obliczeniowej, nie mające nawet gwarancji uzyskania rozwiązań dopuszczalnych zadań, są asymptotycznie optymalne w średnim przypadku. Wyniki tego typu są w oczywisty sposób odmienne od wyników analizy przypadku najgorszego i dlatego stanowią ich wartościowe uzupełnienie.

Pogląd, że analiza przypadku średniego może zastąpić analizę przypadku

najgorszego jest w odczuciu autora błędny. Zarówno analiza przypadku najgorszego, jak również analiza przypadku średniego mają swoją specyfikę oraz uzyskują wartościowe wyniki i oceny. Dopiero zapoznanie się z wynikami analiz różnego rodzaju pozwala wyrobić sobie możliwie najbardziej obiektywny pogląd na różne aspekty analizowanego zadania lub algorytmu optymalizacji dyskretnej.

Należy również podkreślić, że wyniki analizy przypadku średniego są prawdziwe tylko dla rozważanych losowych klas zadań. Należy więc zachować szczególną ostrożność przy próbach uogólniania uzyskanych wyników na inne klasy zadań, gdyż może to prowadzić do fałszywych i nieuzasadnionych wniosków.

Istotnym wyzwaniem badawczym pozostaje nadal przeprowadzenie analizy przypadku średniego dla szczególnie trudnych zadań optymalizacji dyskretnej. Dobrym przykładem jest zadanie programowania całkowitoliczbowego, patrz podrozdział 1.2 oraz wzór (1.2). W przypadku zadania programowania całkowitoliczbowego postać funkcji Lagrange'a oraz zadania dualnego nie sprzyja zastosowaniu technik i oszacowań wykorzystanych w podrozdziałach 5.2 oraz 6.2.

Innym ważnym celem przyszłych prac badawczych wydaje się rozważenie bardziej realistycznych modeli zadań, co w szczególności może wymagać zastosowania złożonych rozkładów prawdopodobieństwa zmiennych losowych opisujących charakterystyki analizowanych zadań. Również w tym przypadku oczekiwane jest uzyskanie analitycznych wyników o podobnym charakterze jak wyniki zawarte w podrozdziałach 5.5 oraz 6.3 niniejszej monografii.

Literatura

[1] R. Aboudi, K. Jörnsten. Tabu search for general zero-one integer programs using the pivot and complement heuristic. *ORSA Journal on Computing*, 6:82–93, 1994.

[2] A.V. Aho, J.E. Hopcroft, J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[3] G. d' Atri. Probabilistic analysis of the knapsack problem. Working Paper 7, Groupe de Récherche 22, Centre National de la Récherche Scientifique, Paris, 1978.

[4] G. Ausiello, A. Marchetti-Spaccamela, M. Protasi. Probabilistic analysis of the solution of the knapsack problem. W: *Proceedings of the Tenth IFIP Conference*, ss. 557–565, New York, 1982. Springer.

[5] I. Averbakh. Probabilistic properties of the dual structure of the multidimensional knapsack problem and fast statistically effcient algorithms. *Mathematical Programming*, 65:311–330, 1994.

[6] E. Balas. An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 13:517–546, 1965.

[7] E. Balas, C.H. Martin. Pivot and complement - a heuristic for 0-1 programming. *Management Science*, 26:86–96, 1980.

[8] E. Balas, E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28:1130–1154, 1980.

[9] R. Battiti, G. Tecchiolli. Local search with memory: benchmarking RTS. *OR Spektrum*, 17:67–86, 1995.

[10] J. Beardwood, J. Halton, J.M. Hammersley. The shortest path through many points. *Proc. Cambridge Philos. Soc.*, 55:299–327, 1959.

[11] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[12] M. Bertocchi, L. Słomiński, J. Sobczyńska. Probabilistic and deterministic local search for solving the binary multiknapsack problem. *Optimization*, 33:155–166, 1995.

[13] J. Błażewicz, K.H. Ecker, E. Pesch, G. Schmidt, J. Węglarz. *Scheduling Computer and Manufacturing Processes*. Springer Verlag, Berlin, Heidelberg, 1996.

[14] J. Błażewicz, J.K. Lenstra, A.H.G. Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5:11–24, 1983.

[15] K.H. Borgwardt. The average number of steps required by simplex-method is polynomial. *Zeitschrift für Operations Research*, 26:157–177, 1982.

[16] I.N. Bronsztejn, K.A. Siemiendiajew. *Matematyka, poradnik encyklopedyczny*. Państwowe Wydawnictwo Naukowe, Warszawa, 1973.

[17] A.V. Cabot. An enumeration algorithm for knapsack problems. *Operations Research*, 18:306–311, 1970.

[18] P.M. Camerini, F. Maffioli, C. Vercellis. Multi-constrained matroidal knapsack problems. *Mathematical Programming*, 45:211–231, 1989.

[19] V. Cerny. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Preprint, Institute of Physics and Biophysics, Comenius University, Bratislava, 1982.

[20] L.G. Chaczian. Wielomiananowy algorytm programowania liniowego. *Dokl. Akad. Nauk SSSR, Nova Seria*, 244(5):1093–1096, 1979. (w języku rosyjskim).

[21] I. Charon, O. Hudry. The noising method: a new method for combinatorial optimization. *Operations Research Letters*, 14:133–137, 1993.

[22] P.C. Chu, J.E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998.

[23] E.G. Coffman Jr, M.R. Garey, D.S. Johnson. Approximation algorithm for bin-packing - an updated survey. W: G. Ausiello, M. Lucertini, P. Serafini, editors, *Algorithm Design for Computer System Design*, ss. 49–106, New York, 1984. Springer.

[24] E.G. Coffman Jr, G.S. Lueker. *Probabilistic Analysis of Packing and Partitioning Algorithms.* Wiley-Interscience, New York, Chichester, Brisbane, Toronto, Singapore, 1991.

[25] E.G. Coffman Jr, G.S. Lueker, A.H.G. Rinnooy Kan. Asymptotic methods in the probablistic analysis of sequencing and packing heuristics. *Management Science*, 34:266–290, 1988.

[26] S.A. Cook. The complexity of theorem-proving procedures. W: *Proc. 3rd Annu. ACM Symp. On Theory of Computing*, ss. 151–158, New York, 1971. ACM Press.

[27] T.H. Cormen, C.E. Leiserson, R.L. Rivest. *Wprowadzenie do algorytmów.* Wydawnictwa Naukowo Techniczne, Warszawa, 1997.

[28] Y. Crama, J.B. Mazzola. On the strength of relaxations of multidimensional knapsack problems. *INFOR*, 32:219–225, 1994.

[29] F. Dammeyer, S. Voss. Dynamic tabu list management using reverse elimination method. *Annals of Operations Research*, 41:31–46, 1993.

[30] G.B. Dantzig. Discrete variable problems. *Operations Research*, 5:266–277, 1957.

[31] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, 1963.

[32] A. Drexl. A simulated annealing approach to the multiconstraint zero-one knapsack problem. *Computing*, 40:1–8, 1988.

[33] K. Dudziński, K. Szkatuła. A note on sequencing jobs with deadlines problem. *European Journal of Operational Research*, 59:333–336, 1992.

[34] G. Dueck. New optimization heuristics. *Journal of Computational Physics*, 104:86–92, 1993.

[35] G. Dueck, T. Schuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90:161–175, 1990.

[36] A.E. Eiben, E.H.L. Aarts, K.H. Van Hee. Global convergence of genetic algorithms: A markov chain analysis. *Lecture Notes in Computer Science*, 496:4–9, 1991.

[37] H. Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417, 1963.

[38] W. Feller. *Wstęp do rachunku prawdopodobieństwa, tom II.* Państwowe Wydawnictwo Naukowe, Warszawa, 1981.

[39] J.F. Fontanari. A statistical analysis of the knapsack problem. *Journal of Physics A - Mathematical and General*, 28:4751–4759, 1995.

[40] G.E. Fox, G.D. Scudder. A heuristic with tie breaking for certain 0-1 integer programming models. *Naval Research Logistics Quarterly*, 32:613–623, 1985.

[41] A. Freville, G. Plateau. Heuristics and reduction methods for multiple constraints 0-1 linear programming problems. *European Journal of Operational Research*, 24:206–215, 1986.

[42] A. Freville, G. Plateau. Hard 0-1 multiknapsack test problems for size reduction methods. *Investigacion Operativa*, 1:251–270, 1990.

[43] A. Freville, G. Plateau. An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem. *Discrete Applied Mathematics*, 49:189–212, 1994.

[44] A. Freville, G. Plateau. The 0-1 bidimensional knapsack problem: toward an efficient high-level primitive tool. *Journal of Heuristics*, 2:147–167, 1997.

[45] A.M. Frieze, M.R.B Clarke. Approximation algorithms for the m-dimensional 0-1 knapsack problem: Worst case and probabilistic analysis. *European Journal of Operational Research*, 15:100–109, 1984.

[46] M.R. Garey, R.L. Graham, D.S. Johnson, A. Yao. Resource constrained scheduling as generalized bin packing. *J. Combin. Theory A*, (21):257–298, 1976.

[47] M.R. Garey, D.S. Johnson. Strong NP-completeness results: motivation, examples and implications. *Journal of the Association of Computer Machinery*, 25:499–508, 1978.

[48] M.R. Garey, D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[49] R.S. Garfinkel, G.L. Nemhauser. *Programowanie całkowitoliczbowe.* Państwowe Wydwnictwo Naukowe, Warszawa, 1978.

[50] S.L. Gass. *Programowanie liniowe.* PWN, Warszawa, 1976.

[51] B. Gavish, H. Pirkul. *Allocation of Databases and Processors in a Distributed Computing System.* North-Holland, Amsterdam, 1982.

[52] B. Gavish, H. Pirkul. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. *Mathematical Programming*, 31:78–105, 1985.

[53] S. Van de Geer, L. Stougie. On rates of convergence and asymptotic normality in the multiknapsack problem. *Mathematical Programming*, 51:349–358, 1991.

[54] P.C. Gilmore, R.E. Gomory. The theory and computation of knapsack functions. *Operations Research*, 14:1045–1075, 1966.

[55] F. Glover. Heuristic for integer programming using surrogate constraints. *Decision Sciences*, 8:156–160, 1977.

[56] F. Glover. Tabu search: a tutorial. *Interfaces*, 20(4):74–94, 1991.

[57] F. Glover. Optimization by ghost image processes in neural networks. *Computers and Operations Research*, 21:801–822, 1994.

[58] F. Glover, G.A. Kochenberger. Critical event tabu search for multidimensional knapsack problems. W: I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, ss. 407–427. Kluwer Academic Publishers, 1996.

[59] F. Glover, M. Laguna. *Tabu Search.* Kluwer, Boston, 1997.

[60] A.V. Goldberg, A. Marchetti-Spaccamela. On finding the exact solutions of a 0-1 knapsack problem. W: *Proceedings of the 16th ACM Symposium on Theory of Computing*, ss. 359–368, New York, 1984. Association for Computing Machinery.

[61] M.R. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[62] M. Grötschel, L. Lovász. Combinatorial optimization. W: R. Graham, M. Grötschel, L. Lovász, editors, *Handbook of Combinatorics*, ss. 1541–1597. Elsevier Science B.V., 1995.

[63] S. Hanafi, A. Freville. An efficient tabu search approach for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, to appear.

[64] S. Hanafi, A. Freville, A.El. Abedellaoui. Comparison of heuristics for the 0-1 multidimensional knapsack problem. W: I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, ss. 449–465. Kluwer Academic Publishers, 1996.

[65] D. Hausman, R. Kannan, B. Korte. Exponential lover bounds on a class of knapsack algorithms. *Mathematics of Operations Research*, 6:225–232, 1981.

[66] F.S. Hiller. Efficient heuristic procedures for integer linear programming with an interior. *Operations Research*, 17:600–637, 1969.

[67] D.S. Hochbaum. A nonlinear knapsack problem. *Operations Research Letters*, 17:103–110, 1995.

[68] A. Hoff, A. Løkketagen, I. Mittet. Genetic algorithms for 0/1 multidimensional knapsack problems. Working paper, Molde College, Britvein 2, Molde, Norway, 1996.

[69] J.H. Holland. Adaptation in natural and artificial systems. Technical report, The University of Michigan Press, Ann Arbor, 1975.

[70] D.S. Johnson. *Near-optimal allocation algorithms.* PhD thesis, MIT, Cambridge, MA, 1973.

[71] D.S. Johnson. The NP completeness column: an ongoing guide. *J. Algorithms*, 5:284–299, 1984.

[72] D.S. Johnson. The NP-completeness column: an ongoing guide. *J. Algorithms*, 9:426–444, 1988.

[73] D.S. Johnson. Local optimization and the travelling salesman problem. W: *Proc. 17th Coll. On Automata, Languages and Programming*, ss. 446–461, Heidelberg, 1990. Springer.

[74] D.S. Johnson. The NP-completeness column: an ongoing guide. *J. Algorithms*, 13:502–524, 1992.

[75] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operational Research*, 37:865–892, 1989.

[76] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon. Optimization by simulated annealing; an experimental evaluation, part II, graph coloring and number partitioning. *Operations Research*, 39:378–406, 1991.

[77] R. Kannan, B. Korte. Approximative combinatorial algorithms. W: *Mathematical Programming*, ss. 195–248, Amsterdam, New York, 1984. North Holland.

[78] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:375–395, 1984.

[79] R.M. Karp. Reducibility among combinatorial problems. W: R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, ss. 85–103. Plenum Press, New York, 1972.

[80] R.M. Karp. The probablilistic analysis of some combinatorial search algorithms. W: J.F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, ss. 1–19. Academic Press, 1976.

[81] R.M. Karp. Probabilistic analysis of partitioning algorithms for the travelling salesman problem in the plane. *Mathematics of Operations Research*, 2:209–224, 1977.

[82] R.M. Karp. A patching algorithms for the nonsymmetric travelling salesman problem. *SIAM Journal on Computing*, 8:561–573, 1979.

[83] R.M. Karp, J.K. Lenstra, C.J.H. McDiarmid, A.H.G. Rinnooy Kan. Probablistic analysis of combinatorial algorithms. W: M. OhEigeartaigh, J.K. Lenstra, A.G.H. Rinnooy Kan, editors, *Combinatorial Optimization: Annotated Bibliographies*, ss. 52–88. Wiley-Interscience, New York, Chichester, 1985.

[84] S. Khuri, T. Bäck, J. Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. W: *Proceedings of the 1994 ACM Symposium on Applied Computing (SAC'94)*, ss. 188–193. ACM Press, 1994.

[85] S. Kirkpatrick, C.D. Gelatt, M. Vecchi. Optimization by simmulated annealing. *Science*, 220:621–680, 1983.

[86] V. Klee, G.J. Minty. How good is the simplex algorithm. W: O. Shisba, editor, *Inequalities III*, ss. 159–175. Academic Press, 1972.

[87] G.A. Kochenberger, B.A. McCarl, F.P. Wyman. A heuristic for general integer programming. *Decision Sciences*, 5:36–44, 1974.

[88] B. Korte, D. Hausmann. An analysis for the greedy algorithm for independence systems. *Ann. Discrete Math.*, 2:65–74, 1978.

[89] B. Korte, L. Lovász, R. Schrader. *Greedoids.* Springer, Heidelberg, 1991.

[90] J.B. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proc. Amer. Math. Soc.*, 2:48–50, 1956.

[91] J.L. Kulikowski. *Zarys Teorii Grafów.* Państwowe Wydawnictwo Naukowe, Warszawa, 1986.

[92] E.L. Lawler, J.M. Moore. A functional equation and its application to resource alocation and sequencing problems. *Management Science*, 16:77–84, 1969.

[93] J.S. Lee, M. Guignard. An approximate algorithm for multidimensional zero-one knapsack problems - a parametric approach. *Management Science*, 34:402–410, 1988.

[94] T.-E. Lee, G.-T. Oh. The asymptotic value-to-capacity ratio for the multi-class stochastic knapsack problem. *European Journal of Operational Research*, 103:584–594, 1997.

[95] L.A. Levin. Average case complete problems. *SIAM J. Comput.*, 15:285–286, 1986.

[96] M. Loève. *Probability Theory I.* Springer Verlag, New York, Heidelberg, Berlin, 1977.

[97] A. Løkketagen, K. Jörnsten, S. Storøy. Tabu search within a pivot and complement framework. *International Transactions of Operations Research*, 1:305–316, 1994.

[98] A. Løkketangen, F. Glover. Probabilistic move selection in tabu search for zero-one mixed integer programming problems. W: I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics*, ss. 467–487. Kluwer Academic Publishers, 1996.

[99] A. Løkketangen, F. Glover. Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research*, to appear.

[100] G.S. Lorie, L. Savage. Three problems in capital rationing. *Journal of Business*, 28:229–239, 1955.

[101] R. Loulou, E. Michaelides. New greedy-like heuristics for the multidimensional 0-1 knapsack problem. *Operations Research*, 27:1101–1114, 1979.

[102] G.S. Lueker. On the average difference between the solution to linear and integer knapsack problems. W: *Applied Probability-Computer Science, the Interface, Vol 1*, ss. 489–504. Birkhauser, Basel, 1982.

[103] M.J. Magazine, O. Oguz. A heuristic algorithm for the multidimensional zero-one knapsack problem. *European Journal of Operational Research*, 16:319–326, 1984.

[104] J.W. Mamer, K.E. Schilling. On the growth of random knapsacks. *Discrete Applied Mathematics*, 28:223–230, 1990.

[105] S. Martello, P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* Wiley & Sons, 1990.

[106] M. May, K. Szkatuła. On the bipartite crossing number. *Control and Cybernetics*, 17:85–98, 1988.

[107] M. Meanti, A.H.G. Rinnooy Kan, L. Stougie, C. Vercellis. A probabilistic analysis of the multiknapsack value function. *Mathematical Programming*, 46:237–247, 1990.

[108] M. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller. Equation of state calculations by fast computing machines. *J. Chemical Physics*, 21:1087–1092, 1953.

[109] G.L. Nemhauser, Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15:494–505, 1969.

[110] G.L. Nemhauser, L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons Inc., New York, 1988.

[111] C.H. Papadimitriou, K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, 1982.

[112] R.G. Parker, R.L. Rardin. *Discrete Optimization.* Academic Press, Boston, 1988.

[113] H. Pirkul. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics*, 34:161–172, 1987.

[114] C.N. Potts, L.N. Van Wassenhove. Algorithms for scheduling a single machine to minimize the weighted number of late jobs. *Management Science*, 34:843–858, 1988.

[115] A.H.G. Rinnooy Kan. Probabilistic analysis of algorithms. *Annals of Discrete Mathematics*, 31:365–384, 1987.

[116] A.H.G. Rinnooy Kan, L. Stougie. Probabilistic analysis of algorithms. W: J.K. Lenstra, H. Tijms, T. Volgenant, editors, *Twenty Five Years of Operations Research in the Netherlands*, ss. 104–121. Math. Centrum Wish. Inform, Amsterdam, 1989.

[117] A.H.G. Rinnoy Kan, L. Stougie, C. Vercellis. A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete Applied Mathematics*, 42:279–290, 1993.

[118] G. Rudolph, J. Sprave. A cellular genetic algorithm with self-adjusting acceptance threshold. W: *Proceedings of the First IEE/IEEE Internatinal Confernce on Genetic Algorithms in Engineering Systems: Innovations and Applications*, ss. 365–372, London, 1995. IEE.

[119] G. Rudolph, J. Sprave. Significance of locality and selection pressure in the grand deluge evolutionary algorithm. W: H.M. Voigt, W. Ebeling, I. Rechenberg, H.P. Schwefel, editors, *Parallel Problem Solving from Nature IV. Proceedings of the International Conference on Evolutionary Computation*, ss. 686–694. Springer, Lecture Notes in Computer Science, 1996.

[120] S.K. Sahni. Algorithms for scheduling independent jobs. *Journal of ACM*, 23:116–127, 1976.

[121] K.E. Schilling. The growth of m-constraint random knapsacks. *European Journal of Operational Research*, 46:109–112, 1990.

[122] K.E. Schilling. Random knapsacks with many constraints. *Discrete Applied Mathematics*, 48:163–174, 1994.

[123] S. Senju, Y. Toyoda. An approach to linear programming with 0-1 variables. *Management Science*, 15:196–207, 1968.

[124] W. Shih. A branch and bound method for the multiconstraint zero-one knapsack problem. *Journal of the Operational Research Society*, 30:369–378, 1979.

[125] S. Smale. On the average number of steps of the simplex method of lineaar programming. *Mathematical Programming*, 27:241–262, 1983.

[126] A.L. Soyster, B. Lev, W. Slivka. Zero-one programming with many variables and few constraints. *European Journal of Operational Research*, 2:195–201, 1978.

[127] J.M. Steele. Probabilistic and worst case analyses of classical problems of combinatorial optimization in Euclidean space. *Marhematics of Operations Research*, 15(4):749–770, 1990.

[128] J.M. Steele. *Probability Theory and Combinatorial Optimization.* Society for Industrial and Applied Mathematics, Philadephia, 1996.

[129] H.I. Stern, Z. Avivi. The selection and scheduling of textile orders with due dates. *European Journal of Operational Research*, 44:11–16, 1990.

[130] A. Straszak, M. Libura, J. Sikorski, D. Wagner. Computer-assisted constrained approval voting. *Group Decision and Negotiation*, 2:375–385, 1993.

[131] M.M. Sysło, N. Deo, J.S. Kowalik. *Discrete Optimization Algorithms.* Prentice-Hall Inc., Englewood Cliffs, 1983.

[132] K. Szkatuła. Probabilistic analysis of the sequencing jobs with deadlines problem and the threshold algorithm. W: G.Menga and V.Kempe, editors, *Proceedings of the Workshop on Informatics in Industrial Automation*, ss. 113–124, Berlin, 1989.

[133] K. Szkatuła. Analiza probabilistyczna wielowymiarowych, ograniczonych całkowitoliczbowych zadań załadunku. *Techniczeskaja Kibernetika*, 4:236–241, 1994. (w języku rosyjskim).

[134] K. Szkatuła. On the asymptotical growth of multi-constraint entirely random knapsacks. W: *Systems Analysis and Decision Support in Economics and Technology*, ss. 299–304, Warszawa, 1994.

[135] K. Szkatuła. On the growth of entirely random multi-constraint 0-1 knapsacks. W: *BOS 93, Trzecia Konferencja Badań Operacyjnych I Systemowych*, ss. 205–304, Warszawa, 1994.

[136] K. Szkatuła. On the growth of multi-constraint random knapsacks with various right-hand sides of the constraints. *European Journal of Operational Reserch*, 73:199–204, 1994.

[137] K. Szkatuła. Analiza średniego przypadku m-wymiarowych zadań załadunku. W: *Wspomaganie Decyzji, Systemy Eksperckie*, ss. 195–200, Warszawa, 1995.

[138] K. Szkatuła. The growth of multi-constraint random knapsacks with large right-hand sides of the constraints. *Operations Research Letters*, 21:25–30, 1997.

[139] K. Szkatuła. The growth of multi-constraint random knapsacks with mixed right-hand-sides of the constraints. Instytut Badań Systemowych PAN, Warszawa, 1997.

[140] K. Szkatuła. Random sequencing jobs with deadlines problem: growth of the optimal solutions values. *European Journal of Operational Research*, 109:160–169, 1998.

[141] K. Szkatuła, M. Libura. Probabilistic analysis of simple algorithms for binary knapsack problem. *Control and Cybernetics*, 12:147–157, 1983.

[142] K. Szkatuła, M. Libura. On probabilistic properties of greedy like algorithms for the binary knapsack problem. W: *Stochastics in Combinatorial Optimization*, Singapore, 1987. World Scientific Publishing.

[143] J. Thiel, S. Voss. Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms. *INFOR*, 32:226–242, 1994.

[144] Y. Toyoda. A simplified algorithm for obtaining approximate solutions to zero-one programming problems. *Management Science*, 21:1417–1427, 1975.

[145] A. Volgenant, J.A. Zoon. An improved heuristic for multidimensional 0-1 knapsack problems. *Journal of the Operational Research Society*, 41:963–970, 1990.

[146] S. Walukiewicz. *Programowanie dyskretne.* Państwowe Wydawnictwo Naukowe, Warszawa, 1986.

[147] B. Weide. *Statistical methods in algorithm design and analysis.* PhD thesis, Carnegie-Mellon University, Pittsbutgh, PA, 1978. CMU-CS 78-142.

[148] H.M. Weingartner. *Mathematical Programming and the Analysis of Capital Budgeting Problems.* Markham Publishing, Chicago, 1967.

[149] H.M. Weingartner, D.N. Ness. Methods for the solution of the multidimensional 0/1 knapsack problem. *Operations Research*, 15:83–103, 1967.

[150] S.H. Zanakis. Heuristic 0-1 linear programming: an experimental comparison of three methods. *Management Science*, 24:91–104, 1977.

[151] K. Zorychta, W. Ogryczak. *Programowanie liniowe i całkowitoliczbowe: metoda podziału i ograniczeń.* Wydawnictwa Naukowo Techniczne, Warszawa, 1981.

Analiza przypadku średniego w optymalizacji dyskretnej

Wielowymiarowe zadanie załadun oraz zadanie szeregowania prac

Krzysztof Szkatuła

W monografii omawiane są wybrane zadania optymalizacji dyskretnej i metody ich rozwiązywania oraz problematyka analizy złożoności obliczeniowej zadań i algorytmów.

Głównym celem książki jest wykazanie, na przykładzie wielowymiarowego zadania załadunku i zadania szeregowania prac, że przy zastosowaniu niezbyt zaawansowanego aparatu rachunku prawdopodobieństwa można uzyskać wartościowe wyniki analizy przypadku średniego w optymalizacji dyskretnej.

Wyniki zaprezentowane w monografii potwierdzają, że analiza przypadku średniego jest bardzo efektywnym narzędziem wspomagającym i uzupełniającym powszechnie stosowane do analizy zadań i algorytmów optymalizacji dyskretnej podejście przypadku najgorszego.

ISBN 83-85847-39-1
ISSN 0208-8029

W celu uzyskania bliższych informacji i zakupu dodatkowych egzemplarzy prosimy o kontakt z Instytutem Badań Systemowych PAN
ul. Newelska 6, 01-447 Warszawa
tel. 37-35-78 w. 241 e-mail: kotuszew@ibspan.waw.pl