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 2

Metody rozwiązywania

2.1 Wprowadzenie

W rozdziale 1 przedstawiono różnorodne zadania optymalizacji dyskretnej. W momencie kiedy dokonany zostanie sformalizowany, z matematycznego punktu widzenia, zapis zadania optymalizacji dyskretnej, powstaje potrzeba uzyskania rozwiązania rozważanego zadania. Celem niniejszego rozdziału monografii jest przybliżenie problematyki i metod rozwiązywania zadań optymalizacji dyskretnej.

Podstawowym celem każdej metody rozwiązywania zadania optymalizacji dyskretnej jest wyznaczenie jego rozwiązania optymalnego, które może być oznaczone jako $z_{OPT}(I^*)$. Bardziej szczegółowo, rozwiązanie optymalne zadania rozumiane jest jako:

- wartość liczbowa rozwiązania optymalnego wyrażona przez $z_{OPT}(I^*)$, na przykład optymalna wartość funkcji celu dla zadania programowania całkowitoliczbowego (1.1) lub wartość sumy wag krawędzi dla najkrótszej drogi w grafie itp;

- postać rozwiązania optymalnego I^*, na przykład jeden z wektorów $[x_1^*, x_2^*, \ldots, x_n^*]$, realizujący wartość funkcji celu $z_{OPT}(I^*)$ w przypadku realizacji danych zadania programowania całkowitoliczbowego lub zbiór krawędzi grafu G będący poszukiwaną najkrótszą drogą itp.

Często wymagane jest wyznaczenie całego zbioru rozwiązań optymalnych zadania, to znaczy wszystkich rozwiązań z optymalną wartością funkcji celu. Istnieją również sytuacje, w których wystarczy wyznaczyć samą wartość rozwiązania optymalnego bez wyznaczania jego postaci, co w wielu przypadkach może okazać się znacznie prostsze.

Nieformalnie rzecz ujmując, metody rozwiązywania zadań optymalizacji dyskretnej można podzielić na *uniwersalne*, to znaczy takie, których główna idea działania może być wykorzystana do rozwiązywania wielu różnorodnych zadań, oraz *wyspecjalizowane*, to znaczy takie, które wykorzystują tak dalece specyficzne właściwości problemu, do rozwiązywania którego zostały zaprojektowane, że nie jest możliwe ich efektywne zastosowanie do rozwiązywania innych zadań.

W dalszej części monografii pojęcia metoda obliczeniowa oraz algorytm będą stosowane zamiennie, z tym, że termin algorytm będzie stosowany w odniesieniu do bardziej skonkretyzowanych metod obliczeniowych. Książka Cormena i innych [27] jest podstawową monografią z zakresu analizy algorytmów.

Algorytm uzyskujący zawsze rozwiązania optymalne zadania będziemy nazywać *algorytmem dokładnym*. Jeżeli przyjęte zostało założenie o skończoności zbioru rozwiązań dopuszczalnych, to wyznaczenie rozwiązania optymalnego zadania jest kwestią przeszukania całego zbioru rozwiązań dopuszczalnych i wybrania z nich najlepszego w sensie wartości kryterium optymalizacyjnego. Metody stosujące strategie tego typu nazywane są metodami pełnego przeglądu.

Niestety, dla dużej części zadań optymalizacji dyskretnej nawet o niezbyt dużych rozmiarach zastosowanie w praktyce obliczeniowej metod pełnego przeglądu jest niewykonalne. Jako przykładu można użyć wektora binarnego, $x = [x_1, x_2, \ldots, x_n]$, gdzie $x_i = 0$ lub 1, $i = 1, 2, \ldots, n$. Wektor x może stanowić rozwiązanie pewnego zadania binarnego, patrz podrozdział 1.3, dla ustalenia uwagi zadania załadunku (1.6). Liczba wszystkich takich wektorów wynosi 2^n. Funkcja 2^n rośnie bardzo szybko wraz ze wzrostem n. Dla $n = 20$ jej wartość przekracza 1 milion (10^6), natomiast dla $n = 40$ przekracza ona 1 bilion (10^{12}). Takie rozmiary zadań są niewielkie przy ich wykorzystaniu dla modelowania zagadnień praktycznych, ale nakład obliczeń niezbędny do rozwiązania zadania załadunku tą metodą, to znaczy wygenerowanie wszystkich wektorów x, sprawdzenie ich dopuszczalności i wybranie spośród nich optymalnego, będzie wymagało nieakceptowalnego nakładu obliczeń. Jeszcze dobitniej rozważania te ilustruje zadanie komiwojażera. W grafie pełnym o n wierzchołkach mamy $\frac{1}{2}(n-1)!$ możliwych cykli Hamiltona. Dla zadania o rozmiarze $n = 30$ nakład obliczeń niezbędnych do przejrzenia i obliczenia wartości wszystkich cykli przekraczałby czas istnienia superkomputera wykonującego te obliczenia. Z tej przyczyny duże wysiłki badaczy zostały poświęcone opracowaniu metod redukujących nakład niezbędnych obliczeń.

Dla wielu zadań optymalizacji dyskretnej istnieją efektywne algorytmy dokładne. Niestety okazało się, że dla wielu istotnych w zastosowaniach praktycznych zadań optymalizacji dyskretnej nie są znane algorytmy dokładne

pozwalające efektywnie, to znaczy przy "akceptowalnym" czasie obliczeń, je rozwiązywać. Pojęcie "akceptowalny" czas obliczeń nie jest precyzyjne, ale można przyjąć, że miesiące, lata lub stulecia pracy obecnych superkomputerów są nie do przyjęcia chociażby ze względów praktycznych. Na przykład badacz rozpoczynający eksperyment obliczeniowy może utracić zainteresowanie jego wynikami lub nawet nie doczekać jego zakończenia.

Ta sytuacja spowodowała burzliwy rozwój metod wyznaczających rozwiązania przybliżone zadań, zwanych dalej *metodami przybliżonymi*. Dla praktycznej użyteczności metod przybliżonych istotna jest wiedza, jak bardzo wartość uzyskanego rozwiązania przybliżonego odbiega od wartości rozwiązania optymalnego. Wprowadzenie miar oceniających jakość metody przybliżonej A wymaga przyjęcia pewnych założeń.

- Zakładamy, że rozważane zadanie optymalizacji dyskretnej w istotny sposób zależy od n parametrów, gdzie n jest liczbą całkowitą, $n \geq 1$. W odniesieniu do zadań na grafach można przyjąć, że n jest liczbą wierzchołków w grafie, $n = |V|$. W odniesieniu do zadania optymalizacji dyskretnej w postaci (1.5) możemy przyjąć, że $n = q$. Aby było możliwe przeprowadzenie analizy asymptotycznej, zakładamy, że wartość n jest nieograniczona, $n \to \infty$.

- Jako I_n oznaczamy konkretną postać rozważanej realizacji danych zadania o n parametrach. Dla grafu ważonego o n wierzchołkach oznacza to, że znany jest zbiór jego krawędzi lub łuków E oraz, że każdemu łukowi lub krawędzi $e \in E$ przypisana jest waga $c(e)$ o znanej wartości liczbowej. W odniesieniu do zadania optymalizacji dyskretnej w postaci (1.5) oznacza to, że zadany jest $\mathcal{F}$: postać zbioru podzbiorów $Q = \{1, \ldots, q\}$ oraz, że składowym wektora $[c_1, \ldots, c_q]$ przypisane są konkretne wartości liczbowe.

- Jako I_n^* oznaczymy rozwiązanie optymalne zadania I_n. Wartość rozwiązania optymalnego oznaczymy jako $z_{OPT}(I_n^*)$, natomiast wartość rozwiązania przybliżonego zadania I_n uzyskanego przez algorytm przybliżony A oznaczymy jako $z_A(I_n)$.

Jako miarę dokładności działania metody przybliżonej A można zastosować iloraz $z_A(I_n)$ oraz wartości rozwiązania optymalnego $z_{OPT}(I_n^*)$. Z lewej strony w (2.1) przedstawiono postać tej oceny odpowiednią dla zadania optymalizacji dyskretnej z kryterium minimalizacyjnym, natomiast z prawej strony dla zadania z kryterium maksymalizacyjnym:

$$\psi(I_n) = \frac{z_{OPT}(I_n^*)}{z_A(I_n)} \qquad \text{lub} \qquad \psi(I_n) = \frac{z_A(I_n)}{z_{OPT}(I_n^*)}. \tag{2.1}$$

Dokładność pracy algorytmu przybliżonego A jest oceniana poprzez najbardziej niekorzystną wartość tego ilorazu dla wszystkich realizacji danych rozpatrywanego zadania z zadanej klasy. Przyjmując założenie, że

$$z_{OPT}(I_n^*),\ z_A(I_n) \geq 0 \text{ oraz } \max\{z_A(I_n), z_{OPT}(I_n^*)\} > 0 \tag{2.2}$$

mamy

$$0 \leq \psi(I_n) \leq 1.$$

W ogólnym przypadku im wartość $\psi(I_n)$ jest bliższa 1 tym wyższa jest dokładność algorytmu A. Jeśli istnieje stała $\epsilon > 0$ taka, że dla wszystkich realizacji danych zadania

$$\epsilon \leq \psi(I_n) \leq 1 \tag{2.3}$$

lub, w przypadku asymptotycznym przy $n \to \infty$, jeżeli $\psi(I_n)$ dąży do pewnej wartości granicznej, to wtedy możemy mówić o *gwarantowanej dokładności* rozważanej metody. Co więcej, jeżeli istnieją $\epsilon_n > 0$ dla $n \geq 1$, takie, że

$$\epsilon_n \leq \psi(I_n) \leq 1 \text{ oraz } \lim_{n\to\infty} \epsilon_n = 1 \tag{2.4}$$

to będziemy mówić, że algorytm przybliżony A jest *algorytmem asymptotycznie (sub)optymalnym*.

Natomiast przy braku takich ocen algorytm nie ma żadnej gwarantowanej dokładności i jest nazywany *algorytmem heurystycznym* (*heuristic algorithm*). Innym istotnym zagadnieniem jest, czy algorytm gwarantuje uzyskanie rozwiązania dopuszczalnego dla wszystkich realizacji danych rozważanego zadania.

Ocena dokładności działania algorytmu A w sensie (2.1) jest oparta na obserwacji zachowania *błędu względnego* danego przez $\psi(I_n)$.

Inną miarą oceny dokładności działania algorytmu jest *błąd bezwzględny* przyjmujący poniższą postać:

$$\gamma(I_n) = |z_A(I_n) - z_{OPT}(I_n^*)|. \tag{2.5}$$

W przypadku oceny w sensie błędu absolutnego jeżeli istnieją $\delta_n \geq 0$ takie, że zawsze $\gamma(I_n) \leq \delta_n$ oraz jeżeli istnieje stała δ taka, że $\delta_n \leq \delta$, to możemy mówić o gwarantowanej dokładności algorytmu A. Jeżli natomiast $\lim_{n\to\infty} \delta_n = 0$, to możemy mówić o asymptotycznej (sub)optymalności algorytmu A. Oceny te są analogicznie do ocen (2.3) oraz (2.4) w sensie błędu względnego.

W ogólnym przypadku, jeżeli spełnione jest (2.2), mamy:

$$\gamma(I_n) = 0 \text{ wtedy i tylko wtedy, kiedy } \psi(I_n) = 1.$$

W przypadku spełnienia powyższych zależności ($\gamma(I_n) = 0$ lub $\psi(I_n) = 1$) A jest algorytmem dokładnym.

Można zauważyć, że miara dokładności pracy algorytmu w sensie (2.1) nosi odmienny charakter niż miara w sensie (2.5). Dla ustalenia uwagi rozważmy zadanie na minimum. Przechodząc do rozważań o charakterze asymptotycznym, gdy $n \to \infty$, mamy:

- Z faktu, że $\lim_{n\to\infty} \psi(I_n) = 1$, nie wynika, że zawsze $\lim_{n\to\infty} \gamma(I_n) = 0$.

Jako przykład możemy przyjąć $z_A(I_n) = n + \sqrt{n}$ oraz $z_{OPT}(I_n^*) = n$. Wtedy $\lim_{n\to\infty} \psi(I_n) = 1$ oraz $\gamma(I_n) \to +\infty$.

- Z faktu, że $\lim_{n\to\infty} \gamma(I_n) = 0$, nie wynika, że zawsze $\lim_{n\to\infty} \psi(I_n) = 1$.

Jako przykład mogą służyć $z_A(I_n) = \frac{1}{n} + \frac{1}{\sqrt{n}}$ oraz $z_{OPT}(I_n^*) = \frac{1}{n}$. Wtedy $\lim_{n\to\infty} \gamma(I_n) = 0$ oraz $\lim_{n\to\infty} \psi(I_n) = 0$.

Mówiąc ogólnie, ocena w sensie (2.1) ignoruje człony asymptotycznie wolniej rosnące niż $\max\{z_A(I_n), z_{OPT}(I_n^*)\}$. Jeżeli zachodzi (2.2) oraz przy założeniu, że $\max\{z_A(I_n), z_{OPT}(I_n^*)\} \geq c$, gdzie $c > 0$ stała, stwiedzenie powyższe może być wyrażone w sposób następujący:

$$\text{Jeżeli } \frac{\gamma(I_n)}{\max\{z_A(I_n), z_{OPT}(I_n^*)\}} \to 0, \text{ to wtedy } \psi(I_n) \to 1. \tag{2.6}$$

Zarysowane powyżej podejście do oceny dokładności pracy algorytmów jest nazywane *analizą przypadku najgorszego* (*worst case analysis*), to znaczy dokładność działania algorytmu jest oceniana przez najmniej korzystny wynik uzyskany dla wszystkich realizacji danych zadania. W rozdziale 3 przedstawimy podejście przypadku najgorszego do oceny złożoności obliczeniowej (niezbędnego nakładu obliczeń) zadań i algorytmów optymalizacji dyskretnej. Natomiast w rozdziale 4 przedstawimy alternatywne podejście analizy przypadku średniego do oceny zadań i algorytmów optymalizacji dyskretnej, które w rozdziałach 5 oraz 6 zastosujemy do wybranych zadań i algorytmów.

W kolejnych podrozdziałach zaprezentujemy wybrane optymalne oraz przybliżone metody rozwiązywania zadań optymalizacji dyskretnej.

2.2 Metoda podziału i oszacowań

Jak już wspomniano wcześniej, z powodu nadmiernego nakładu obliczeń metoda pełnego przeglądu zbioru rozwiązań dopuszczalnych nie może być stosowana do rozwiązywania zadań optymalizacji dyskretnej o większych, praktycznych rozmiarach.

Idea *metody podziału i oszacowań (branch and bound)* polega na podziale oryginalnego zadania na podproblemy oraz na wygenerowniu dobrych oszacowań wartości rozwiązania optymalnego zadania dla tych podproblemów w celu znacznego zredukowania ilości kroków przeglądu. Pominięte są podproblemy nie rokujące poprawy wartości funkcji celu. Dla ustalenia uwagi załóżmy, że rozważamy zadanie optymalizacji dyskretnej w postaci danej przez (1.5). Zakładamy, że:

- Zadanie ma skończony zbiór rozwiązań dopuszczalnych $\mathcal{F}$.
- Potrafimy w efektywny sposób *podzielić* zbiór rozwiązań dopuszczalnych $\mathcal{F}$ na podzbiory, na przykład, gdzie $\bigcup_{i=1}^{k} \mathcal{F}_i = \mathcal{F}$ oraz $\mathcal{F}_i \bigcap \mathcal{F}_j = \emptyset$, $i \neq j$, $i, j \in \{1, \ldots, k\}$. Niech $\mathcal{F}_0 = \mathcal{F}$. Uzyskujemy w ten sposób zbiór podproblemów zadania oryginalnego (1.5), gdzie zbiór rozwiązań dopuszczalnych $\mathcal{F}$ został zastąpiony przez podzbiór $\mathcal{F}_i$ w postaci:

$$z_{OPT}(q) = \max_{F \in \mathcal{F}_i} z(F). \tag{2.7}$$

  Zakładamy też, że podobną procedurę możemy zastosować do dowolnego podzbioru $\mathcal{F}$.

- Potrafimy konstruować *relaksację* (osłabienie) zadania (2.7) w postaci:

$$\bar{z}_{OPT}(q) = \max_{F \in \Phi_i} \bar{z}(F) \tag{2.8}$$

  gdzie $i = 0, 1, \ldots,$ $\mathcal{F}_i \subseteq \Phi_i$ oraz $z(F) \leq \bar{z}(F)$ dla $F \in \mathcal{F}_i$.

- Na początku pracy algorytmu dysponujemy pewnym oszacowaniem wartości rozwiązania optymalnego zadania (1.5) $z(F^*)$, to znaczy

$$z(F^*) \leq z_{OPT}(q).$$

  Oszacowanie to może być dane przez pewne znane nam rozwiązanie dopuszczalne zadania (1.5) F^* lub przyjmujemy $z(F^*) = -\infty$. W przypadku $z(F^*) = -\infty$ postać F^* jest nieokreślona.

Metoda podziału i oszacowań działa według poniższych zasad:

1. Dokonujemy początkowego podziału zbioru rozwiązań dopuszczalnych zadania na k, $k \geq 2$, podzbiorów $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$, które zapamiętujemy jako listę podproblemów do rozważenia oraz znamy wartość $z(F^*)$.

2. W ogólnym kroku pracy algorytmu rozważamy kolejny podproblem z listy: $\mathcal{F}_i$. Uzyskujemy rozwiązanie optymalne F' jego relaksacji (2.8). Mogą teraz zaistnieć trzy sytuacje:

   - $z(F^*) \geq \bar{z}(F')$. Wtedy w $\mathcal{F}_i$ nie istnieje rozwiązanie o wartości większej niż $z(F^*)$ a więc można ten podproblem usunąć z listy i wykluczyć z dalszych rozważań.
   - $z(F^*) < \bar{z}(F')$ oraz $F' \in \mathcal{F}_i$. Wtedy podstawiamy $F^* := F'$. F' jest rozwiązaniem optymalnym podproblemu (2.7). Oznacza to, że ten podproblem nie zawiera lepszych rozwiązań zadania (1.5) niż F' a więc może on zostać usunięty z listy i wykluczony z dalszych rozważań.
   - Jeżeli $z(F^*) < \bar{z}(F')$ oraz $F' \notin \mathcal{F}_i$ to wtedy nie istnieje możliwość rozstrzygnięcia o perspektywiczności $\mathcal{F}_i$. Dokonujemy podziału $\mathcal{F}_i$ na dodatkowe podproblemy, które dołączamy do listy podproblemów i kontynuujemy działanie metody.

3. Metoda podziału i oszacowań kończy pracę, kiedy lista podproblemów jest pusta. Rozwiązaniem optymalnym zadania (1.5) jest F^*, a jego wartością $z(F^*)$. Jeżeli zadanie nie ma rozwiązań dopuszczalnych, to metoda zakończy pracę z nieokreśloną wartością F^* oraz $z(F^*) = -\infty$.

Niewątpliwą zaletą tej metody jest fakt, że dla każdej realizacji danych uzyskuje ona rozwiązanie optymalne zadania lub stwierdza, że zadanie nie posiada rozwiązań dopuszczalnych w skończonej liczbie kroków. Natomiast nakład wykonanych obliczeń zależy od jakości stosowanych relaksacji podproblemów zadania oryginalnego. Jeżeli dodatkowo dysponujemy "dobrym" początkowym rozwiązaniem dopuszczalnym zadania F^* o wartości funkcji celu $z(F^*)$ "bliskiej" do wartości rozwiązania optymalnego (1.5) $z_{OPT}(q)$ to w takim przypadku możemy oczekiwać znacznego zmniejszenia nakładu obliczeń w wyniku efektywniejszej eliminacji podproblemów z listy.

W wariancie niekorzystnym metoda podziału i oszacowań może okazać się tożsama z metodą pełnego przeglądu. W przypadku zadania załadunku oznacza to taki podział na podproblemy, że zostaną rozważone wszystkie 2^n różne wektory binarne $x = [x_1, x_2, \ldots, x_n]$.

Dla zadań programowania całkowitoliczbowego, patrz na przykład (1.2), relaksację podproblemów można uzyskać pomijając warunek całkowitoliczbowości zmienych decyzyjnych, to znaczy poprzez zastąpienie odpowiedniego zadania programowania całkowitoliczbowego zadaniem programowania liniowego. Metoda podziału i oszacowań została szczegółowo omówiona w monografii Zorychty i Ogryczaka [151].

2.3 Programowanie dynamiczne

Programowanie dynamiczne jest ogólną techniką opracowaną przez Bellmana, patrz [11], pierwotnie dla optymalnego podejmowania decyzji w systemach mogących się znajdować w ograniczonej liczbie *stanów*. Podstawową zasadą tej techniki jest *zasada optymalności*, która głosi, że

> "optymalny ciąg decyzji ma tę właściwość, że jakikolwiek byłby stan i decyzje początkowe, pozostałe decyzje muszą stanowić optymalny ciąg decyzji w odniesieniu do stanu wynikającego z decyzji początkowych".

Praktycznie każde z zadań optymalizacji dyskretnej może być rozwiązywane przez metody oparte na idei programowania dynamicznego. Uzyskane w oparciu o zasadę programowania dynamicznego algorytmy, dla poszczególnych zadań optymalizacji dyskretnej, są metodami rekurencyjnymi wykorzystującymi zadania pomocnicze. Niestety, algorytmy takie są efektywne tylko wtedy, kiedy liczba zadań pomocniczych jest niewielka. Działanie tej metody zostanie przedstawione poniżej na przykładzie *jednowymiarowego zadania załadunku* (1.6), patrz też podrozdział 5.6 oraz (5.47). Dla konstruowania zadań pomocniczych wygodnie jest rozważyć zadanie (1.6) w poniższej postaci:

$$z_{OPT}(k,b) = \max \sum_{i=1}^{k} c_i \cdot x_i$$
$$\text{przy ograniczeniu} \quad \sum_{i=1}^{k} a_i \cdot x_i \leq b$$
$$\text{gdzie } x_i = 0 \text{ lub } 1,\ i = 1,\ldots,k.$$

Przyjmujemy założenie, że k, c_i, a_i oraz b są nieujemnymi liczbami całkowitymi. Zadanie w postaci (1.6) możemy z łatwością uzyskać podstawiając $k = n$, $b = b(n)$. Wartość rozwiązania optymalnego zadania (1.6) jest dana przez $z_{OPT}(n, b(n))$. Z uwagi na binarność zmiennych decyzyjnych, $x_i = 0$ lub 1, dla zadania (1.6) mamy:

$$z_{OPT}(n, b(n)) = \max \{z_{OPT}(n-1, b(n)),\ z_{OPT}(n-1, b(n) - a_n) + c_n\}.$$

W ogólnym przypadku możemy rozważyć następującą zależność rekurencyjną:

$$z_{OPT}(j+1, b) = \max \{z_{OPT}(j, b),\ z_{OPT}(j, b - a_{j+1}) + c_{j+1}\} \tag{2.9}$$

gdzie $j = 0,\ldots,n-1$, $z_{OPT}(0, b) = 0$. Rozwiązanie optymalne $z_{OPT}(n, b(n))$ możemy uzyskać stosując (2.9) kolejno dla wszystkich $j = 0,\ldots,n-1$ oraz $0 < b \leq b(n)$.

Liczba rozważanych zadań pomocniczych zależy od wartości n a konkretniej od wartości $b(n)$. Dla małych wartości n oraz $b(n)$ metoda programowania dynamicznego jest efektywna obliczeniowo, natomiast wraz ze wzrostem wartości n oraz $b(n)$ efektywność tej metody obliczeniowej maleje. Dla "dużych" wartości $b(n)$, to znaczy "bliskich" $\sum_{i=1}^{n} a_i$, staje się ona praktycznie tożsama z metodą pełnego przeglądu.

Karp w pracy [81] zaproponował ciekawe zastosowanie metody programowania dynamicznego do rozwiązywania zadania komiwojażera na płaszczyźnie, patrz podrozdział 1.4, strona 35.

Rozważa on następujący wariant zadania komiwojażera. Dany jest graf ważony G, którego zbiór wierzchołków $V = \{1, \ldots, n\}$ jest określony jako zbior n punktów w kwadracie znajdującym się na płaszczyźnie. Możemy przyjąć, że boki kwadratu są równe 1, bez ograniczenia ogólności rozważań. Każdy z wierzchołków grafu $i \in V$ może zostać zdefiniowany poprzez podanie jego współrzędnych (x_i, y_i), gdzie $0 \leq x_i, y_i \leq 1$. Dla każdej krawędzi lub łuku $e = (i, j)$ grafu G waga c_{ij} jest zdefiniowana jako najkrótsza odległość pomiędzy wierzchołkami. Przyjmuje ona postać:

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.10}$$

Zastosowanie metody programowania dynamicznego dla tej postaci zadania komiwojażera w grafie o n wierzchołkach wymaga rozważenia $n^2 \cdot 2^n$ zadań pomocniczych, co prowadzi do bardzo dużego nakładu obliczeń.

Karp zaproponował, aby jednostkowy kwadrat podzielić na około $n/\log n$ mniejszych kwadratów tak, aby liczba wierzchołków w każdym z nich nie przekraczała znacząco $\log n$. W każdym z tak uzyskanych kwadratów rozwiązujemy odpowiednio mniejsze zadania komiwojażera z zastosowaniem programowania dynamicznego. Wymaga to dla wszystkich mniejszych kwadratów rozważenia około n^4 zadań pomocniczych, co bardzo ogranicza niebędny nakład obliczeń. Uzyskane w mniejszych kwadratach rozwiązania zadań komiwojażera są łączone, aby uzyskać rozwiązanie przybliżone zadania pierwotnego o wartości $Z_{PAR}(n)$. Metoda ta nosi nazwę *algorytm podziału Karpa* (*Karp partiniong algorithm*)

W sensie oceny (2.3) algorytm podziału Karpa nie ma żadnych gwarancji dokładności działania. Natomiast w przypadku średnim algorytm podziału jest asymptotycznie suboptymalny, patrz podrozdział 4.4.

2.4 Podejścia zachłanne

Jednym z bardziej naturalnych podejść do budowy algorytmu dla rozwiązywania zadań optymalizacyjnych są *metody zachłanne* (*greedy*) oparte na

zasadzie konstruowania rozwiązania zadania poprzez dokonywanie najlepszych *lokalnych* wyborów. Dobrym przykładem ilustrującym tą zasadę dla zadania wyznaczenia najkrótszego drzewa rozpinającego, które zostało zaprezentowane w podrozdziale 1.4, jest *algorytm Kruskala*, patrz [90].

Idea podejścia zachłannego polega w tym przypadku na wyznaczeniu krawędzi $e_1, e_2, \ldots, e_{n-1}$ w $n-1$ iteracjach. W każdej iteracji wyznaczana jest jedna krawędź należąca do najkrótszego drzewa rozpinającego nieskierowanego grafu ważonego G. W pierwszej iteracji wyznaczana jest krawędź o najmniejszej wadze, to znaczy:

- $e_1 : c(e_1) = \min\limits_{e \in A} c(e)$.

W iteracjach $k = 2, 3, \ldots, n-1$ wyznaczamy w rekurencyjny sposób kolejne krawędzie e_k według następujących zasad:

- $e_k : c(e_k) = \min\limits_{e \in A_k} c(e)$ gdzie

$$A_k = \{e \in A : e \notin \{e_1, e_2, \ldots, e_{k-1}\} \text{ i } \{e_1, e_2, \ldots, e_{k-1}, e\} \text{ jest lasem}\}$$

W każdej iteracji algorytmu dokonywany jest najlepszy lokalny wybór, dlatego algorytm nazywany jest zachłannym. Uzyskane rozwiązanie zadania wyzaczenia najkrótszego drzewa rozpinającego jest zawsze optymalne. Struktura grafu G nie odgrywa praktycznie większej roli w działaniu algorytmu Kruskala, wymagamy jedynie, aby podgraf grafu G składający się ze zbioru krawędzi $\{e_1, e_2, \ldots, e_{k-1}, e\}$ oraz wierzchołków je tworzących był lasem w celu uniknięcia dołączenia do konstruowanego rozwiązania krawędzi tworzących cykl.

Rozważmy teraz zadania optymalizacyjne (1.12), (1.13) oraz (1.14) zdefiniowane dla zadania rozbicia zbioru, patrz podrozdział 1.3. Bez ograniczenia ogólności możemy przyjąć założenie, że wagi $x_1, x_2, \ldots, x_n$ wprowadzone na stronie 24 są posortowane w porządku nierosnącym, a więc

$$x_1 \geq x_2 \geq \ldots \geq x_n. \tag{2.11}$$

Jak zauważono na stronie 25, przy $m = 2$ zadania (1.12) oraz (1.13) są ze sobą tożsame w sensie postaci rozwiązań optymalnych. Dla ustalenia uwagi rozważmy zadanie (1.13) przy $m = 2$. Do jego rozwiązywania zastosujmy algorytm działający według poniższej zasady:

Element i, $i = 1, \ldots, n$, jest przydzielany do zbioru N_1, jeżeli i jest liczbą nieparzystą oraz do zbioru N_2, jeżeli i jest liczbą parzystą.

Uzyskane rozwiązanie przybliżone ma wartość funkcji celu $z_{GRE}(n)$ spełniającą poniższe zależności, patrz Rinnooy Kan [115] oraz Rinnooy Kan i Stougie [116]:

$$z_{GRE}(n) = \max\left\{\sum_{i\in N_1} x_i, \sum_{i\in N_2} x_i\right\} \text{ oraz } \frac{2}{3} \leq \frac{z_{OPT}(n,2)}{z_{GRE}(n)} \leq 1 \quad (2.12)$$

Dla zadania pakowania zasobników (1.14), gdzie (2.11) jest spełnione, dużą popularnością cieszą się dwa algorytmy zachłanne:

- *Pierwszy pasujący malejący* (*first fit decreasing* w skrócie FFD). Każdy kolejny element i, $i = 1, \ldots, n$, jest pakowany do zasobnika o najmniejszym numerze, w którym się on mieści. Niech wartość uzyskanego przez algorytm FFD rozwiązania (liczba wykorzystanych zasobników) będzie oznaczona $z_{FFD}(n)$.

- *Najlepiej pasujący malejący* (*best fit decreasing* w skrócie BFD). Każdy kolejny element i, $i = 1, \ldots, n$, jest pakowany do zasobnika o najmniejszej ilości wolnego miejsca, w którym się on mieści. Niech wartość uzyskanego przez algorytm BFD rozwiązania (liczba wykorzystanych zasobników) będzie oznaczona $z_{BFD}(n)$.

Spełnienie warunku (2.11) powoduje "zachłanność" powyższych metod. Rozważmy algorytmy *pierwszy pasujący* (*first fit* w skrócie FF) oraz *najlepiej pasujący* (*best fit* w skrócie BF), które działają dokładnie tak samo jak algorytmy FFD oraz BFD opisane powyżej z tą różnicą, że warunek (2.11) nie jest spełniony. Oznaczmy wartości rozwiązań przybliżonych (ilość zużytych zasobników) przez nie uzyskiwanych odpowiednio jako $z_{FF}(n)$ oraz $z_{BF}(n)$. Dla algorytmów FF oraz FFD znane są następujące zależności, patrz Johnson [70] oraz Garey i inni [46],

$$z_{FF}(n) \leq \left\lceil \tfrac{17}{10} \cdot z_{OPT}(n) \right\rceil \text{ oraz } z_{FFD}(n) \leq \left\lceil \tfrac{11}{9} \cdot z_{OPT}(n) \right\rceil + 4 \quad (2.13)$$

gdzie $\lceil a \rceil$ oznacza najmniejszą liczbę całkowitą taką, że $a \leq \lceil a \rceil$. Co więcej, dla każdego z tych algorytmów istnieją przykłady zadań takich, że

$$\left\lceil \tfrac{17}{10} \cdot z_{OPT}(n) \right\rceil - 1 \leq z_{FF}(n) \text{ oraz } \left\lceil \tfrac{11}{9} \cdot z_{OPT}(n) \right\rceil \leq z_{FFD}(n).$$

Dla algorytmów BF oraz BFD znane są podobne wyniki, patrz Coffman i inni [23]. Wyniki te wykazują dobitnie, że wprowadzenie elementu zachłanności w znaczący sposób poprawia jakość uzyskiwanych rozwiązań. W podrozdziale 4.4 zaprezentowane zostaną wyniki analizy przypadku średniego w odniesieniu do zadania pakowania zasobników.

Dla wybranych zadań optymalizacji dyskretnej podejście zachłanne pozwala sformułować algorytmy dokładne. Jest to ściśle związane z pojęciem *matroidów*.

Matroidy i algorytm zachłanny

Niech $N = \{1, \ldots, n\}$ będzie skończonym zbiorem, $[c_1, \ldots, c_n]$, $c_i \geq 0$ będzie wektorem *wag* o n składowych. Dla podzbioru E zbioru N, $E \subseteq N$, niech

$$z(E) = \sum_{i \in E} c_i$$

będzie wartością funkcji celu. Niech $\mathcal{E}$ będzie rodziną podzbiorów zbioru N taką, że

jeżeli $X \in \mathcal{E}$ oraz $Y \subseteq X$, to wtedy również $Y \in \mathcal{E}$.

Parę $(N, \mathcal{E})$ nazywamy *systemem niezależności* (*independence system*). Zadanie optymalizacyjne może być wtedy zdefiniowane w postaci:

$$z_{OPT}(n) = \max_{E \in \mathcal{E}} z(E)$$

Algorytm zachłanny dla zadania (2.4) w pierwszej iteracji wyznacza element o najmniejszej wadze, to znaczy:

$$e_1 : c_{e_1} = \min_{e \in N} \{c_e : \exists X \in \mathcal{E}, e \in X\}; \quad z_{GRE}(n) := c_{e_1}.$$

W kolejnych iteracjach wyznaczamy w rekurencyjny sposób kolejne elementy e_k według następujących zasad:

$$e_k : c_{e_k} = \min_{e \in N_k} c_e; \quad z_{GRE}(n) := z_{GRE}(n) + c_{e_k}$$

gdzie

$$N_k = \{e \in N : e \notin \{e_1, e_2, \ldots, e_{k-1}\} \text{ oraz } \{e_1, e_2, \ldots, e_{k-1}, e\} \in \mathcal{E}\}$$

Algorytm zachłanny kończy pracę, jeśli nie istnieje kolejny element e_k. Liczba wykonanych przez niego iteracji n^* nie przekracza n, $n^* \leq n$. Wartość uzyskanego rozwiązania wynosi $z_{GRE}(n)$.

Zdefiniujmy dla $X \subseteq N$ następujące wielkości:

$$\begin{aligned} r(X) &= \max\{|Y| : Y \subseteq X, Y \in \mathcal{E}\} \\ \rho(X) &= \min\{|Y| : Y \subseteq X, Y \in \mathcal{E}, \nexists U \in \mathcal{E} \text{ takie, że } Y \subset U \subseteq X\} \end{aligned}$$

Jeżeli dla wszystkich $X \subseteq N$ zachodzi $r(X) = \rho(X)$, to wtedy system niezależności $(N, \mathcal{E})$ jest nazywany *matroidem*, patrz Grötschel i Lovász [62], Kannan i Korte [77] oraz Korte i Hausmann [88].

Zbiory Q, F, $\mathcal{F}$ w (1.4) i (1.5) są odpowiednikami zbiorów N, X, $\mathcal{E}$. W sformułowania zadania optymalizacji dyskretnej w postaci (1.5) nie wymagamy, aby $(Q, \mathcal{F})$ było systemem niezależności. W pracy Korte i Hausmann [88] udowodniono, że algorytm zachłanny dla zadania (1.5) z dowolnym wektorem wag $[c_1, \ldots, c_q]$, $c_i \geq 0$ uzyskuje rozwiązanie optymalne wtedy i tylko wtedy, kiedy $(Q, \mathcal{F})$ jest matroidem.

W ogólnym przypadku, jeżeli $(N, \mathcal{E})$ jest systemem niezależności, mamy:

$$\min_{X \subseteq N} \left\{ \frac{\rho(X)}{r(X)} \right\} \leq \frac{z_{GRE}(n)}{z_{OPT}(n)} \leq 1$$

co dla licznych zadań optymalizacji dyskretnej pozwala uzyskać gwarancje dokładności dla algorytmu zachłannego. Niestety, dla dużej części zadań optymalizacji dyskretnej algorytm zachłanny pozostaje metodą heurystyczną. Jest on bardzo często używany w praktyce obliczeniowej dla uzyskania "rozsądnych" rozwiązań, na przykład w celu uzyskania rozwiązań przybliżonych w metodzie podziału i oszacowań.

W pracy Korte i inni [89], zdefiniowano pojęcie *greedoidów* (greedoids) oraz przedstawiono klasy zadań optymalizacji dyskretnej, dla których algorytm zachłanny jest optymalny.

W rozdziale 5 niniejszej monografii omówiono działanie algorytmu zachłannego i jego pochodnych dla różnych postaci zadań załadunku.

Zauważmy, że przedstawione powyżej metody podziału i oszacowań oraz programowania dynamicznego dają zawsze rozwiązania optymalne dla rozwiązywanych zadań. Algorytm zachłanny uzyskuje rozwiązania optymalne dla wybranych zadań optymalizacji dyskretnej, a dla pewnej grupy zadań jest on algorytmem przybliżonym o gwarantowanej dokładności. Natomiast w bardzo wielu przypadkach jest to metoda heurystyczna. W kolejnym podrozdziale przedstawimy grupę metod heurystycznych opartych na idei lokalnej poprawy rozwiązań dopuszczalnych.

2.5 Metody lokalnej poprawy i inne

Dla ustalenia uwagi przyjmujemy, że rozważamy zadania optymalizacji dyskretnej w postaci danej przez (1.5). Ogólna idea działania metod lokalnej poprawy może być przedstawiona w poniższej postaci:

- W pierwszym kroku algorytmu uzyskujemy początkowe *bieżące* rozwiązanie przybliżone F^*, $F^* \in \mathcal{F}$, z wartością funkcji celu $z(F^*)$.
- W iteracyjnym kroku algorytmu potrafimy uzyskać rozwiązanie przybliżone F', $F' \in \mathcal{F}$, *bliskie* do F^*. Jeśli $z(F^*) < z(F')$, to wtedy

dokonujemy zamiany bieżącego rozwiązania przybliżonego $F^* := F'$ i przechodzimy do kolejnej iteracji algorytmu.

- Możliwe jest uzyskiwanie nie jednego, a kilku rozwiązań przybliżonych, bliskich do bieżącego. Wtedy najlepsze z nich jest porównywane z rozwiązaniem bieżącym i ewentulanie go zastępuje.

- Algorytm kończy pracę w przypadku braku możliwości dalszej poprawy wartości bieżącego rozwiązania. Ostatnie rozwiązanie bieżące staje się rozwiązaniem przybliżonym uzyskanym przez algorytm lokalnej poprawy.

Sformułowanie *rozwiązanie bliskie* do *bieżącego* jest w ogólnym przypadku bardzo nieprecyzyjne. W praktyce chodzi tutaj o wykonanie stosunkowo nieskomplikowanej obliczeniowo modyfikacji rozwiązania bieżącego zadania tak, aby uzyskać jedno lub kilka rozwiązań przybliżonych. W przypadku zadań sformułowanych jako binarne, to znaczy kiedy F^* odpowiada wektor $[x_1^*, \ldots, x_n^*]$, $x_i^* = 1$ $(i \in F^*)$ lub $x_i^* = 0$ $(i \notin F^*)$, $i = 1, \ldots, n$, rozwiązanie bliskie do bieżącego możemy próbować uzyskiwać poprzez zamianę jednej (lub kilku) składowych: $x'_k \leftarrow 1 - x_k^*$ (dla pozostałych $x'_k \leftarrow x_k^*$), jeżeli ta operacja nie narusza dopuszczalności rozwiązania. W przypadku zadań teorii grafów, dla których rozwiązaniami są zbiory łuków lub krawędzi grafu, na przykład drogi, cykle lub drzewa, rozwiązania bliskie do bieżącego uzyskujemy poprzez zamianę pewnego podzbioru łuków lub krawędzi rozwiązania bieżacego na inne, z zachowaniem dopuszczalności uzyskanego w ten sposób rozwiązania.

Zaletą metod lokalnej poprawy jest spodziewany niski nakład obliczeniowy. Jak już wspomniano, algorytm kończy pracę, jeśli nie jest w stanie poprawić bieżącego rozwiązania. Może to oznaczać wyznaczenie przez algorytm *optimum lokalnego*. Niestety, wartość uzyskanego w ten sposób rozwiązania przybliżonego zadania może być odległa od wartości rozwiązania optymalnego. Aby wyeliminować, albo przynajmniej w istotny sposób złagodzić tę niedogodność zaproponowano szereg modyfikacji metody przeglądu lokalnego próbujących omijać optima lokalne. Część tych metod nosi charakter uniwersalny, na przykład przegląd losowy, symulowane wychładzanie lub przeszukiwanie z zakazami. Metody te mogą być stosowane do szerokich klas zadań optymalizacji dyskretnej. Niestety, dość często nie są one efektywne.

Z kolei takie techniki jak algorytmy genetyczne i inne metody (np. sieci neuronowe) są oparte na analizie konkretnych zadań. Są one ściśle dopasowane do zadań, które rozwiązują i często nie mają waloru ogólności. Możliwości ich zastosowania do innych zadań są więc ograniczone. Poniżej zostaną omówione ogólne zasady działania wymienionych powyżej algorytmów.

Przeszukiwanie losowe

Jedną z prób właściwego ukierunkowania przeszukiwania zbioru rozwiązań dopuszczalnych i takiego sposobu akceptacji rozwiązań, aby zwiększyć prawdopodobieństwo wyznaczenia przez algorytm rozwiązania optymalnego, jest *randomizacja* algorytmów. W przypadku metody lokalnej poprawy randomizacja może być zastosowana w sposób następujący.

- Niech $\varphi(F) = \exp(z(F)/T)$ dla każdego $F \in \mathcal{F}$, gdzie T jest pewną małą liczbą dodatnią. Mamy początkowe *bieżące* rozwiązanie przybliżone F^*, $F^* \in \mathcal{F}$, z wartością funkcji celu $z(F^*)$.

- W iteracyjnym kroku algorytmu potrafimy uzyskać rozwiązanie przybliżone F', $F' \in \mathcal{F}$, *bliskie* do F^*. Jeśli $z(F^*) < z(F')$, to wtedy dokonujemy akceptacji uzyskanego rozwiązania przybliżonego $F^* := F'$. Natomiast jeżeli $z(F') \leq z(F^*)$, to wtedy dokonujemy akceptacji uzyskanego rozwiązania przybliżonego $F^* := F'$ z prawdopodobieństwem $\varphi(F')/\varphi(F^*)$ i krok iteracyjny jest powtarzany.

- Metoda kończy pracę albo w przypadku nie uzyskania poprawy wartości rozwiązania w zadanej ilości iteracji, albo po wykonaniu określonej liczby iteracji.

W pracy Metropolisa i innych [108] udowodniono asymptotyczną optymalność takiego algorytmu. Niestety, z uwagi na bardzo powolną zbieżność praktyczna przydatność tego podejścia jest wątpliwa.

Symulowane wychładzanie

W roku 1983 w dwóch równolegle przedstawionych pracach, Cerny [19] oraz Kirckpatrick i inni [85] zaproponowano metodę rozwiązywania złożonych zadań optymalizacji dyskretnej opartą na analogii do osiągania stanu minimalnej energii materii wychładzanej po rozgrzaniu do stanu topnienia (wrzenia). Podejście to nazwano metodą *symulowanego wychładzania* (*simulated annealing*). W odniesieniu do zadań optymalizacji dyskretnej traktujemy zbiór rozwiązań dopuszczalnych zadania $\mathcal{F}$ jako zbiór stanów ciała (materii) poddawanej wychładzaniu, $1/z(F)$ jest energią stanu (rozwiązania dopuszczalnego) $F \in \mathcal{F}$. Parametr T ma fizyczną interpretację jako temperatura materii (ciała) podlegającej wychładzaniu. Na początku procesu (pracy algorytmu) wartość $T = T^0$ jest duża, równa temperaturze topnienia (wrzenia) materii oraz mamy początkowe *bieżące* rozwiązanie przybliżone F^*, $F^* \in \mathcal{F}$ z wartością funkcji celu $z(F^*)$. Następnie temperatura jest stopniowo zmniejszana

w dyskretnych krokach (mamy $T^1 > T^2 > \cdots > T^k > \cdots$). Dla każdej kolejnej wartości temperatury T^k wykonujemy losowe przeszukiwanie opisane w poprzednim podrozdziale, to znaczy uzyskujemy bliskie do bieżącego rozwiązanie przybliżone zadania F'. W przypadku poprawy wartości rozwiązania jest ono akceptowane, natomiast w przypadku $\varphi(F^*) \geq \varphi(F')$ dokonujemy zamiany (akceptacji F') $F^* := F'$ z prawdopodobieństwem $\varphi(F')/\varphi(F^*)$. Procedurę tę powtarzamy dla kolejnych wartości temperatur. Algorytm kończy pracę, kiedy w zadanej liczbie iteracji nie nastapiła poprawa uzyskanego rozwiązania, albo jeżeli liczba iteracji staje się nadmierna.

Wykorzystując wyniki Metropolisa i innych, patrz [108] można udowodnić, że algorytm symulowanego wychładzania jest asymptotycznie zbieżny do rozwiązania optymalnego zadania z prawdopodobieństwem 1. Pierwsze eksperymenty polegające na zastosowaniu tej metody do szczególnie trudnych zadań optymalizacji dyskretnej, na przykład dla zadania komiwojażera, były bardzo zachęcające. Niestety, bardziej szczegółowa analiza jej działania wykazała, że jest ona bardzo wrażliwa na wybrany schemat wychładzania, to znaczy wartości temperatur $T^1, T^2, \cdots, T^k, \cdots$, i co za tym idzie nie jest stabilna obliczeniowo. Uzykane wyniki eksperymentów obliczeniowych zaprezentowano w wielu publikacjach, na przykład w pracach Johnsona i innych, patrz [75], [73] oraz[76].

Przeszukiwanie z zakazami

W powyżej przedstawionych algorytmach przeszukiwania losowego i symulowanego wychładzania zawarta jest idea losowego omijania optimów lokalnych poprzez akceptację pogorszenia wartości funkcji celu z prawdopodobieństwem $\varphi(F')/\varphi(F^*)$. Idea *metody przeszukiwania z zakazami* (*tabu search*) oparta jest na omijaniu optimów lokalnych w sposób deterministyczny. Podejście to zostało zaproponowane przez Glovera, patrz [55] i [56] oraz Glover i Laguna [59].

Metoda to posiada szereg wariantów, dla ustalenia uwagi przedstawimy ją w poniższej postaci, zwanej metodą *najmniejszego pogorszenia* (least deterioration). Dla opisu działania tej metody niezbędne jest zdefiniowanie $\mathcal{N}(F^*)$, zbioru rozwiązań bliskich do aktualnie rozważanego rozwiązania przybliżonego F^*. W przypadku zadań binarnych, to znaczy kiedy F^* odpowiada wektor $[x_1^*, \ldots, x_n^*]$, $x_i^* = 1$ $(i \in F^*)$ lub $x_i^* = 0$ $(i \notin F^*)$, $i = 1, \ldots, n$, zbiór rozwiązań bliskich do bieżącego $\mathcal{N}(F^*)$ możemy próbować uzyskiwać poprzez zamianę części składowych: $x'_k \leftarrow 1 - x_k^*$ (dla pozostałych $x'_k \leftarrow x_k^*$), jeżeli ta operacja nie narusza dopuszczalności rozwiązania. Drugim ważnym elementem tej metody jest *lista zakazów* (tabu list). Zasadniczym celem jej utworzenia jest uniemożliwienie powrotu do niedawno rozważanych roz-

wiązań, w tym optimów lokalnych, co może doprowadzić do zapętlenia się algorytmu.

- W pierwszym kroku algorytmu uzyskujemy początkowe *bieżące* rozwiązanie przybliżone F^*, $F^* \in \mathcal{F}$ z wartością funkcji celu $z(F^*)$.
- W iteracyjnym kroku algorytmu wybieramy rozwiązanie przybliżone F', $F' \in \mathcal{N}(F^*)$, maksymalizujące $z(F') - z(F^*)$ i nie znajdujące się na liście zakazów. Następnie umieszczamy F^* na liście zakazów oraz dokonujemy zamiany bieżącego rozwiązania przybliżonego $F^* := F'$ i przechodzimy do kolejnej iteracji algorytmu.
- Po określonej liczbie iteracji rozwiązanie jest usuwane z listy zakazów.
- Algorytm kończy pracę w przypadku braku uzyskania poprawy bieżącego rozwiązania w zadanej liczbie iteracji.

Zauważmy, że wartość $z(F') - z(F^*)$ może być zawsze ujemna dla pewnych rozwiązań bieżących, będących optimami lokalnymi. W takim przypadku wybierane jest rozwiązanie pogarszające wartość rozwiązania w najmniejszym stopniu i nie znajdujące się na liście zakazów. Bez zastosowania listy zakazów w następnej iteracji algorytm mógłby wrócić do tego samego optimum lokalnego i nigdy go nie opuścić. Z drugiej strony elastyczność działania algorytmu wymaga, aby po pewnym czasie usunąć z listy zakazów wcześniej umieszczone tam rozwiązania.

Algorytmy genetyczne

Algorytmy genetyczne są rozumiane jako "inteligentne" metody przeszukiwania losowego. Podstawy teoretyczne zostały opracowane przez Hollanda, patrz [69]. Idea algorytmu genetycznego jest oparta na procesie ewolucji organizmów biologicznych w przyrodzie. Jedną z podstawowych zasad ewolucji jest dobór naturalny oraz "przetrwanie najlepiej przystosowanych". Jednostki, które potrafią lepiej przystosować się do środowiska w którym żyją mają lepsze szanse na przetrwanie i rozmnażanie się, pozostałe osobniki zostają wyeliminowane. Oznacza to, że *geny* lepiej dostosowanych jednostek będą przenoszone na rosnącą liczbę osobników we wszystkich kolejnych pokoleniach. Co więcej, kombinacja cech dobrze przystosowanych przodków może spowodować, że potomkowie będą jeszcze lepiej przystosowani do środowiska.

Proces ewolucji genetycznej jest symulowany w pracy algorytmów genetycznych dla zadań optymalizacji dyskretnej. Każdy osobnik (potencjalne *rozwiązanie* zadania optymalizacj dyskretnej) jest zakodowany w postaci

chromosomu (łańcucha). Dostosowanie osobnika (rozwiązania) do środowiska jest oceniane przez wartość funkcji oceniającej. Wysoce przystosowani osobnicy (rozwiązania przybliżone o "dobrej" wartości funkcji oceniającej) mogą się *krzyżować* poprzez wymianę informacji genetycznej z innymi wysoce przystosowanymi osobnikami. W ten sposób powstaje *potomstwo*, to znaczy rozwiązania zadania optymalizacji dyskretnej łączące cechy rozwiązań - *przodków*. Po krzyżowaniu często stosowana jest *mutacja*, mająca na celu zamianę pewnych genów w chromosomach. W efekcie skrzyżowania możliwe są dwa sposoby postępowania:

1. Wymieniona zostanie cała populacja, co odpowiada nowemu podzbiorowi rozwiązań dopuszczalnych zadania.

2. Tylko gorzej dostosowani osobnicy zastaną zamienieni, a więc wyeliminowane zostaną rozwiązania dopuszczalne o gorszych wartościach funkcji oceniającej.

Należy zwrócić uwagę, że przy praktycznej implementacji algorytmu genetycznego dla konkretnego zadania bardzo istotnym problemem jest utrzymanie dopuszczalności rozwiązań w kolejnych krokach pracy algorytmu. Ogólny schemat działania algorytmu może być przedstawiony w poniższy sposób:

- Uzyskujemy początkową populację (zbiór rozwiązań). Obliczamy przystosowanie osobników (rozwiązań).

- W iteracyjnym kroku algorytmu dokonujemy wyboru z populacji przodków, wykonujemy krzyżowanie, mutację i uzyskujemy potomstwo. Obliczamy stopień przystosowania potomstwa (wartości funkcji oceniającej). Dokonujemy zamiany części lub całości populacji przodków na potomstwo (dokonujemy zamiany rozwiązań).

- Algorytm kończy pracę, gdy uzyskane zostało rozwiązanie o satysfakcjonującej jakości, lub po wykonaniu zadanej ilości iteracji.

W pracy Eibena i innych, patrz [36], zostało udowodnione, że proces ten zmierza do uzyskania rozwiązania optymalnego zadania z prawdopodobieństwem dążącym do jedności. Ten wartościowy wynik teoretyczny nie przesądza oczywiście o praktycznej przydatności tego algorytmu do rozwiązywania różnorodnych zadań optymalizacji dyskretnej w praktyce obliczeniowej.

Innymi metodami przybliżonymi stosowanymi do rozwiązywania zadań optymalizacji dyskretnej są algorytmy wykorzystujące sieci neuronowe (neural networks), algorytmy akceptacji progu (threshold accept) i wiele innych.

Metody programowania liniowego

Jak już wspomniano w podrozdziale 1.2, zadanie programowania liniowego (1.3) nie jest zadaniem optymalizacji dyskretnej w ścisłym znaczeniu. Natomiast wiele ważnych, praktycznych zagadnień ekonomicznych, technicznych czy też związanych z badaniami operacyjnymi może zostać sformułowanych wprost w postaci zadań programowania liniowego. Co więcej, odgrywa ono bardzo ważną rolę jako zadanie wspomagające w teorii i praktyce rozwiązywania zadań optymalizacji dyskretnej. Poprzez pominięcie warunku całkowitoliczbowości zmiennych x_i lub zastąpienie zmiennych binarnych ($x_i = 0$ lub 1) zmienną ciągłą ($0 \leq x_i \leq 1$) zadania programowania liniowego stają się w naturaly sposób relaksacjami zadań programowania całkowitoliczbowego lub binarnego. Bardzo często są one wykorzystywane dla uzyskiwania oszacowań rozwiązania optymalnego w metodzie podziału i oszacowań, patrz podrozdział 2.2. Ponadto, mogą one być stosowane jako podzadania w modelowaniu wielu ważnych zagadnień praktycznych. Spowodowało to burzliwy rozwój teorii i praktyki rozwiązywania zadań programowania liniowego, patrz Dantzig [31], Gass [50] oraz Zorychta i Ogryczak [151].

Do rozwiązywania zadania programowania liniowego powszechnie stosowana jest, również w pakietach komercyjnych, *metoda sympleks*. Istnieją różne metody sympleksowe: *pierwotna*, *dualna* itp oraz różne, oparte na nich algorytmy obliczeniowe, na przykład *zrewidowana* metoda sympleks. Podstawową ideą tego podejścia jest przeglądanie *rozwiązań bazowych* zadania, odpowiadających wierzchołkom wielościennego zbioru wypukłego. Przeglądu dokonuje się do momentu uzyskania rozwiązania optymalnego lub stwierdzenia wystąpienia *nieograniczoności* wartości optymalnej rozwiązania zadania programowania liniowego, lub też stwierdzenia nie istnienia rozwiązania optymalngo. Pewną wadą tej metody jest jej wykładnicza złożoność obliczeniowa wykazana przez Klee i Minty w pracy [86], patrz rozdział 3, występująca w praktyce obliczeniowej w bardzo rzadkich, patologicznych przypadkach.

W pracy Chacziana [20] została zaproponowana zupełnie odmienna metoda rozwiązywania zadań programowania liniowego, nazwana *algorytmem elipsoidalnym*. Zasadnicza idea tego podejścia sprowadza się do konstruowania ciągu zawężających się elipsoid zawierających rozwiązania przybliżone zadania. W kolejnych krokach elipsoidy są dzielone na pół hiperpłaszczyzną równoległą do jednego z niespełnionych ograniczeń. Uzyskana połowa elipsoidy jest wykorzystywana do skonstruowania następnej elipsoidy. Algorytm kończy pracę w skończonej liczbie kroków a efektem jego działania jest wyznaczenie rozwiązania optymalnego zadania programowania linowego lub stwierdzenie, że ono nie istnieje. Podstawową zaletą tego podejścia jest niska (wielomianowa) złożoność obliczeniowa algorytmu, co pozwoliło zaliczyć

zadanie programowania liniowego do "prostszych" zadań optymalizacyjnych, patrz rozdział 3. Metoda Chacziana oparta jest na bardzo szczególnym modelu obliczeniowym i w praktyce obliczeniowej okazała się niezbyt przydatna.

Karmarkar w pracy [78] zaproponował *algorytm projekcyjny* (*projective algorithm*) dla rozwiązywania zadań programowania liniowego. Algorytm ten należy do metod punktu wewnętrznego. Podstawowa zasada tego podejścia polega na projekcji wektorów na wielościany, w celu uzyskania następnych wektorów będących lepszym przybliżeniem rozwiązania optymalnego zadania programowania liniowego. Na zakończenie pracy algorytmu uzyskujemy rozwiązanie optymalne zadania, albo stwierdzamy, że ono nie istnieje. W przypadku tej metody oraz jej modyfikacji dokonanych przez innych autorów odnotowano bardzo zachęcające wyniki eksperymentów obliczeniowych.

2.6 Podsumowanie

Celem tego rozdziału było przedstawienie wybranych metod obliczeniowych służących do rozwiązywania zadań optymalizacji dyskretnej. Przeprowadzono klasyfikację metod rozwiązywania zadań optymalizacji dyskretnej. Wyodrębniono metody dokładne, uzyskujące rozwiązanie optymalne zadania dla wszystkich realizacji jego danych oraz metody przybliżone, które uzyskują rozwiązanie dopuszczalne zadania o wartości "bliskiej" do optymalnej. Wśród metod przybliżonych dodatkowo wyodrębniono:

- Metody o gwarantowanej dokładności działania. W przypadku tego typu metod znana jest maksymalna wartość względnego lub bezwzględnego błędu, o który różni się wartość rozwiązania przybliżonego od wartości rozwiązania optymalnego.

- Metody heurystyczne. W przypadku metod heurystycznych nie ma żadnych gwarancji jakości uzyskiwanych rozwiązań. Tym niemniej są one często wykorzystywane w licznych zastosowaniach praktycznych, głównie z powodu małego nakładu obliczeń.

Innym kryterium podziału metod rozwiązywania zadań optymalizacji dyskretnej jest uniwersalność lub wyspecjalizowanie metody. Bardziej szczegółowo zostały omówione następujące podejścia do rozwiązywania zadań optymalizacji dyskretnej:

- Metoda podziału i oszacowań, której podstawowym celem jest zawężenie pełnego przeglądu zbioru rozwiązań dopuszczalnych przy wykorzystaniu oszacowań wartości rozwiązania optymalnego.

- Programowanie dynamiczne, gdzie rozwiązanie optymalne zadań o większym rozmiarze jest uzyskiwane poprzez rozwiązania optymalne mniejszych zadań pomocniczych.

- Metody zachłanne, które mają za zadanie dokonywanie najlepszych lokalnie wyborów. Metody tego typu są często stosowane w praktyce. W niektórych przypadkach można udowodnić ich optymalność, w innych mogą one mieć gwarantowaną dokładność obliczeń. W przypadku dużej części zadań optymalizacji dyskretnej metody zachłanne są metodami heurystycznymi.

- Grupa metod lokalnej poprawy, gdzie zasadnicza idea sprowadza się do uzyskiwania w kolejnych iteracjach, w otoczeniu aktualnie rozpatrywanego rozwiązania przybliżonego zadania, innego rozwiązania przybliżonego, o lepszej wartości funkcji celu. Uzyskane rozwiązanie staje się nowym aktualnym rozwiązaniem. Spośród metod lokalnej poprawy większą uwagę poświęcono przeszukiwaniu losowemu, symulowanemu wychładzaniu, przeszukiwaniu z zakazami oraz algorytmom genetycznym.

- Metody programowania liniowego, które są bardzo użyteczne jako narzędzie pomocnicze w rozwiązywaniu wielu zadań optymalizacji dyskretnej. Omówiono ogólne założenia metody sympleks, algorytmów elipsoidalnego oraz projekcyjnego.

Zamiarem autora było przedstawienie idei działania wybranych metod rozwiązywania zadań optymalizacji dyskretnej. Omówienie, często zawiłych i złożonych, szczegółów technicznych zostało świadomie pominięte w celu zachowania jasności i przejrzystości wywodu. Również przedstawiony wybór omówionych metod jest subiektywny i dalece nie wyczerpujący.

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