FRI-2G.303-1-CCT1-03

DYNAMIC PROGRAMMING VISUALIZER [3]

Assoc. Prof. Emilia Golemanova, PhD
Department of Computer Systems and Technologies,
"Angel Kanchev" University of Ruse
Tel.: 082-888-681
E-mail: EGolemanova@uni-ruse.bg

Assist. Prof. Tzanko Golemanov, PhD
Department of Computer Systems and Technologies,
"Angel Kanchev" University of Ruse
Tel.: 082-888-681
E-mail: TGolemanov@uni-ruse.bg

Abstract: *Dynamic Programming (DP) is a complex and challenging algorithm design technique. This report presents a specialized educational tool designed to help students understand the DP paradigm. A visual tutor has been developed that interactively demonstrates the two fundamental dynamic programming approaches - Bottom-up Approach and Top-Down with Memoization Approach, specifically for 0/1 Knapsack Problem. Students can follow the DP technique step by step, visualizing the DP table as it is populated and observing the transitions between cells. Hovering over each cell provides an explanation of how its value was derived. Additionally, the tool allows users to track the execution of an algorithm by displaying its source code. The visualization system also includes a testing mode, which highlights incorrectly filled cells and offers hints for the correct values and their calculations. By providing visualization and guidance, this educational tutor aids students in effectively learning DP strategies and enables lecturers to focus more on explaining the underlying concepts rather than generating tables in class.*

Keywords: *Dynamic Programming, 0/1 Knapsack Problem, educational tool*

JEL Codes: *C88*

ВЪВЕДЕНИЕ

Въпреки, че алгоритмите са основна и важна част от всяка програма по компютърни науки, студентите често намират курсовете по „Дизайн и анализ на алгоритми" и „Алгоритми и структури от данни" за едни от най-трудните. Поради нейната сложност, областта „алгоритмика" е предизвикателство дори за добрите студенти, които искат да я разберат в нейната пълнота. Отчитайки древната мисъл „една картина струва колкото хиляда думи", много преподаватели считат, че използвайки визуализация, студентите ще разберат алгоритъма по-бързо и по-детайлно (Yi, J. S. et al., 2008).

Според Price et al. (Price, B., Baecker, R. and Small, I., 1998) „визуализация на алгоритъм е визуализацията на абстракциите от по-високо ниво, които описват софтуера". През последните две десетилетия бяха създадени много системи за визуализация или библиотеки, поддържащи различни класове алгоритми (Shaffer, C. A., Cooper, M. and Edwards, S. H., 2007). Някои от тях са алгоритми, работещи върху специфични структури от данни, като дървета, графи или низове. Други системи позволяват илюстриране на алгоритми, които решават конкретна задача, като сортиране, или класове алгоритми, като геометрични алгоритми. Накрая, ограничен е броят на системите, предоставящи визуализации на специфични техники за проектиране на алгоритми, като алчния подход (Velázquez-Iturbide, J. A. Debdi, O., Esteban-Sánchez, N. and Pizarro, C., 2013), Branch-and-Bound алгоритъма (Ramani, K. V. and Rama Rao, T. P., 1994) или метода "разделяй и владей" (Velázquez-Iturbide, J. Á., Pérez-Carrasco, A. and Urquiza-Fuentes, J., 2009).

[3] Докладът е представен на 25 октомври 2024 г. в секция „Комуникационна и компютърна техника" с оригинално заглавие на български език: ВИЗУАЛИЗАТОР ЗА ДИНАМИЧНО ПРОГРАМИРАНЕ.

Техниките (методи, подходи, парадигми) за проектиране на алгоритми са много важни за обучението по алгоритми, тъй като те предлагат общ подход за решаване на задачи алгоритмично, който е приложим към широк кръг проблеми от различни области на компютърното инженерство. Съществува де факто консенсус относно най-важните техники. Добре известни учебници (напр. (Levitin, A., 2012; Alsuwaiyel, M., 2010; Dasgupta S., C. Papadimitriou and U. Vazirani, 2007; Cormen T. H. et al., 2012)) единодушно включват глави, посветени на трите основни техники за проектиране на алгоритъм (а именно „разделяй и владей“, алчни алгоритми и динамично програмиране). Динамичното програмиране е техника за решаване на оптимизационни задачи, която е най-сложната от гореспоменатите. Въпреки това обаче, според Shaffer et al. (Shaffer, C. A., Cooper, M. and Edwards, S. H., 2007) не са много системите за визуализация на този подход за дизайн на алгоритми, които да подпомогнат разбирането на парагигмата на динамичното програмиране.

Докладът представя специално разработения инструмент за визуализация на основните алгоритми за динамично програмиране при решаване на задачата за раницата (*0/1 Knapsack Problem)*, използван за целите на обучението и тестването по дисциплината „Съвременни методи за проектиране на алгоритми“ на студенти от специалност „Компютърни системи и технологии“ към Русенски университет „Ангел Кънчев“.

ИЗЛОЖЕНИЕ

Динамично програмиране (ДП)

Динамичното програмиране, открито от американския математик Ричард Белман през 50-те, е програмистка техника, подобна на „разделяй и владей“, но решаваща задачи с припокриващи се подзадачи. Тя е една от техниките за дизайн на алгоритми, решаващи т. нар. трудни задачи (неполиномиални задачи). Основната идея изисква дефиниране на задачата чрез рекурентна зависимост между припокриващи се подзадачи и използване на таблица с резултатите от вече решени подзадачи, с цел избягване на повторни пресмятания.

Динамичното програмиране е сложна техника – особено за някои задачи като задачата за раницата. Освен това се предлага в два основни варианта. Подходът „отгоре надолу“, обикновено наричан „рекурсивен с мемоизация“, прилага изчисление на откритата рекуренна връзка като рекурсивна функция и добавя памет (ДП-таблица), така че всяка подзадача да се решава само веднъж. Подходът „отдолу нагоре“, обикновено наричан итеративен, по същество изчислява същата рекурентност, но по итеративен начин: дизайнерът на алгоритъма определя ред, в който се обработват подзадачите, и този ред е избран по такъв начин, че всеки път, когато се обработва конкретна задача, оптималните решения на всички нейни необходими подзадачи са вече известни на алгоритъма.

Преподаването на концепцията „динамично програмиране“ е предизвикателство както за обучаващия, така и за обучаемия. Преподавателят се нуждае от удобен инструмент - симулатор, чрез който да обясни алгоритъма за конкретен пример и софтуер, автоматизиращ процеса на тестване на обучаемите. Обучаемите от своя страна имат необходимост от софтуер, автоматизиращ процеса на самообучение (проверка на знания).

Основните изисквания към средството за визуализация на ДП могат да бъдат резюмирани по следния начин:

- представяне на двата подхода за ДП - итеративен и рекурсивен с мемоизация;
- представяне на алгоритъма за генериране на решението на базата на построената ДП-таблица;
- проследяване на попълването на ДП-таблица и „подсказка“ как се изчислява текущата клетка;
- анимационен режим и постъпков режим;
- визуализация на кода на алгоритъма и проследяване на неговото изпълнение.

Съществуващи автоматизирани средства за визуализация на ДП за задачата за раницата (*0/1 Knapsack Problem*)

Съобразно гореспоменатите изисквания бяха анализирани шест съществуващи автоматизирани средства за визуализация на ДП за задачата за раницата и бяха констатирани следните техни недостатъци:

- визуализация само на възходящ подход - (*Knapsack Visualizer*, 2024; *Knapsack Algorithm Visualization*, 2024a; *Algorithm Visualizer*, 2024; *Knapsack Algorithm Visualization*, 2024b)
- няма визуализация на алгоритъма за възстановяване на решението - (*Knapsack Visualizer*, 2024; *Algorithm Visualizer*, 2024; *Knapsack Algorithm Visualization*, 2024b)
- само анимационен режим - (*Knapsack Visualizer*, 2024)
- няма анимационен режим - (*Knapsack Algorithm Visualization*, 2024a; *Knapsack Algorithm Visualization*, 2024b)
- няма „подсказка“ за начина на изчисляване на стойностите в клетките - (*Knapsack Visualizer*, 2024; *Dynamic Programming Visualization*, 2024; *Algorithm Visualizer*, 2024; *Knapsack Algorithm Visualization*, 2024b)
- входът на задачата не е част от таблицата - (*Dynamic Programming Visualization*, 2024; *Algorithm Visualizer*, 2024; *Knapsack Algorithm Visualization*, 2024b)
- липсва код на алгоритъма - (*Knapsack Visualizer*, 2024; *Interactive algorithms. Solving big problems by combining small decisions Dynamic programming. Knapsack 01.*, 2024; *Knapsack Algorithm Visualization*, 2024b)
- ограничения относно размера на екземпляра на задачата - (*Knapsack Algorithm Visualization*, 2024a)
- само фиксиран екземпляр на задачата без възможност за потребителски вход - (*Knapsack Algorithm Visualization*, 2024b)
- липса на допълнителни контроли (скорост на визуализация, пауза на визуализацията, стъпка назад, стъпка напред и др.) - (*Knapsack Visualizer*, 2024; *Knapsack Algorithm Visualization*, 2024a)

Предлагано авторско решение

Предложеният софтуерен продукт е разработен чрез Embarcadero® Delphi в средата на Embarcadero RAD Studio 10.3.3 Rio.

Авторското средство за визуализация на ДП върху задачата за раницата има три основни режима:

- Обучение;
- Самообучение;
- Тест.

Това позволява то да може да бъде използвано както по време на занятия, така и при самоподготовка.

Освен това изборът на език (български, английски, руски) дава възможност за използването му за обучение и на чуждестранни студенти.

Основните функционалности на разработения образователен инструмент са:

- визуализация на трите основни алгоритъма за ДП – итеративен подход, рекурсивен подход с мемоизация и възстановяване на крайното решение на базата на ДП-таблица;
- анимационен или постъпков режим на визуализация с подходящи контроли;
- подходящо оцветяване анализираните клетки и “подсказка” как се изчислява текущата клетка от ДП-таблицата;

- визуализация на кода на алгоритъма и указване на текущо изпълнявания оператор от него;
- избор на екземпляр на задачата:
  - фиксирани примери;
  - потребителски вход;
  - случайно генериран екземпляр.

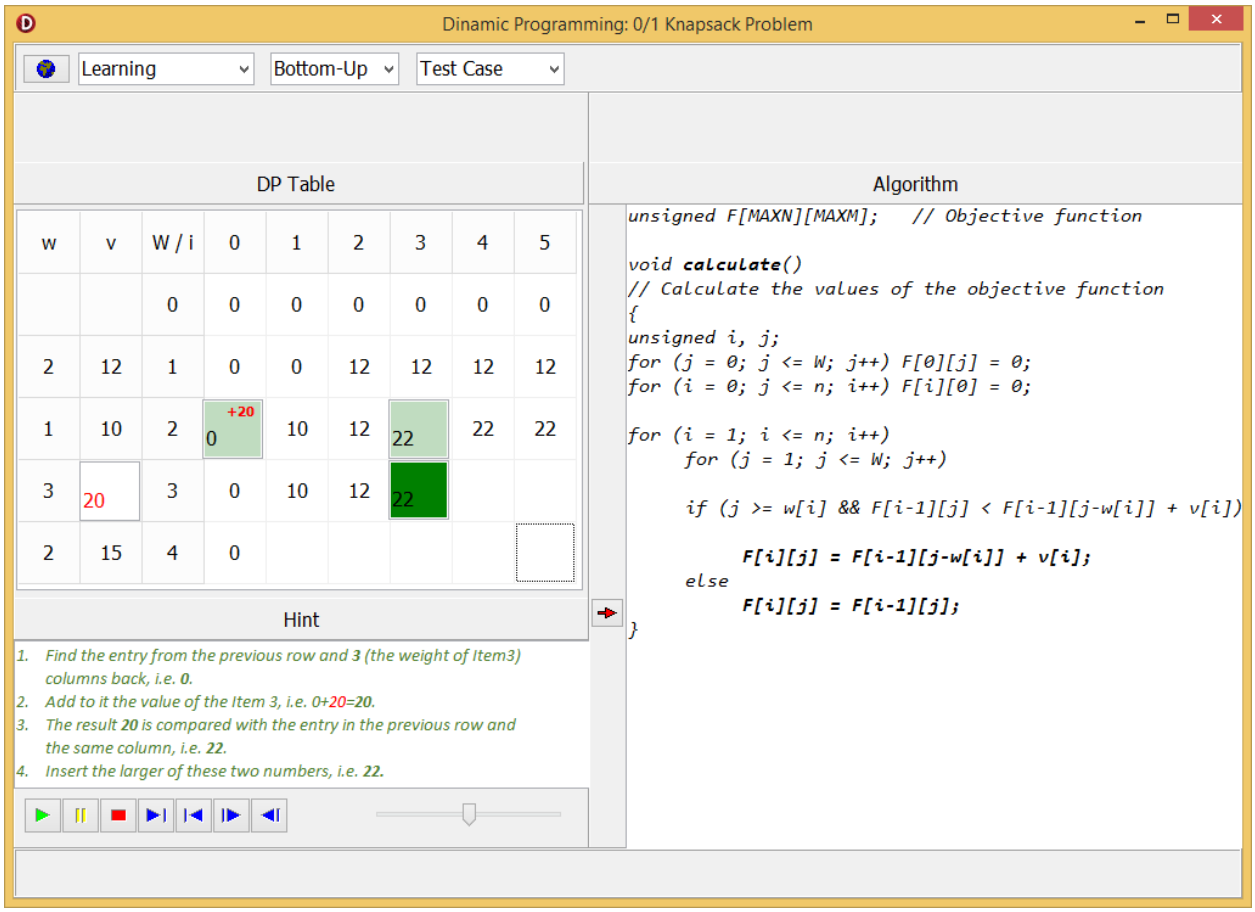
Режим: Обучение

Този режим може да се използва от преподавателя за демостриране на процеса на решаване на задачата за раницата чрез ДП. Това е режимът, при който симулаторът постъпково изпълнява избрания алгоритъм (*Bottom-Up* или *Top-Down*), като автоматично попълва ДП-таблицата (фиг. 1). Атоматичното попълване може да стане в анимационен режим чрез предоставена възможност за пауза, спиране и определяне на скоростта анимацията или постъпково чрез подходящи контроли (*Next/Back, Forward/Backward*). Използвано е подходящо оцветяване на:

- текущо попълваната клетка;
- двете анализирани клетки, определящи стойността на текущата клетка;
- стойността на текущия предмет, която се добавя към предходно изчислена клетка.

Основните стъпки при определяне на стойността на текущо попълваната клетка от таблицата се визуализират в прозореца „*Hint*”.

Интерфейсът на визуализатора представя и кода алгоритъма с указател (стрелка) на текущо изпълнявания оператор от него.

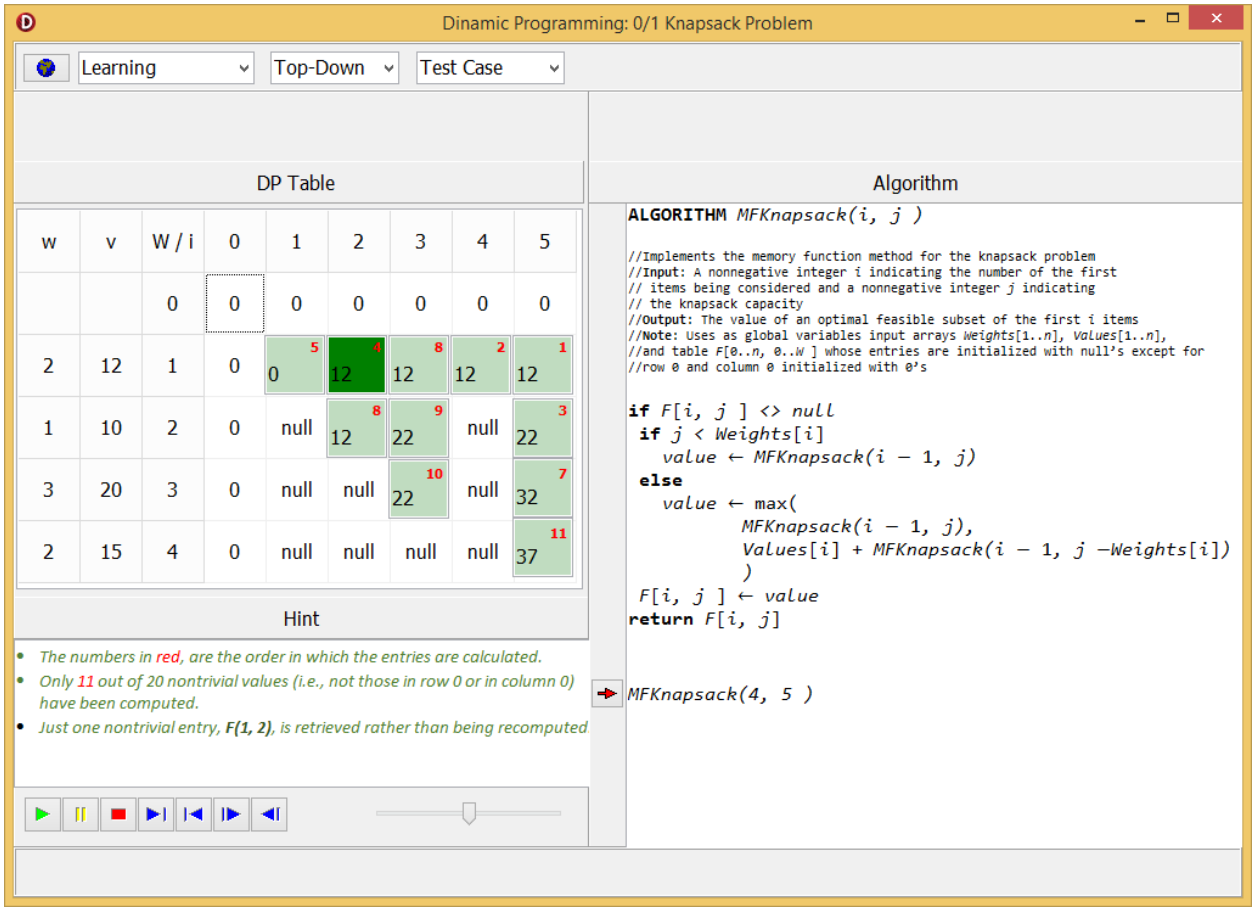


Фиг. 1. Режим „Обучение“: алгоритъм *Bottom-Up*

Основен недостатък на итеративният подход при ДП е, че изисква решаване на всички подзадачи с размер, по-малък от този на изходната задача, което води до излишни пресмятания, тъй като в процеса на пресмятане може да се окаже, че някои от подзадачите не са нужни. Понякога премахването на излишните пресмятания може да доведе до значително ускоряване. В такъв случай се прилага специален вариант на ДП, известен в англоезичната литература като „рекурсивен с мемоизация (*memoization*)“ (или само „мемоизация“), който съчетава предимствата на двата подхода: итеративния и рекурсивния. За целта отново се използва ДП-таблица, която обаче в общия случай не е необходимо да се попълва изцяло. Вместо да пристъпи към директно решаване на някоя подзадача, рекурсивната функция първо търси решението ѝ в таблицата и ако го намери, го взема наготово. Така, подзадачата се решава, само ако никога преди това не е била решавана, след което резултатът незабавно се попълва в таблицата. Решават се само тези подзадачи, които реално могат да участват в оптималното решение.

Рекурсията винаги е била абстрактна и трудна за познаване в дълбочина концепция. Затова използването на разработеното средство за визуализация на рекурсивния подход с мемоизация е особено полезно. Интерфейсът на симулатора при изпълнение на този ДП-алгоритъм (фиг. 2) е подобен на този от Фиг. 1. Разликата е в това, че алгоритъмът е представен чрез псевдо-код и ДП-таблицата на финалната стъпка съдържа непопълнени клетки (отбелязани с *null*). Използвано е подходящо оцветяване на:

- текущо попълваните клетки и реда на тяхното изчисляване (числото в червено в горния десен ъгъл на клетката);
- клетката, чиято стойност се извлича от таблицата вместо отново да бъде изчислена.



Фиг. 2. Режим „Обучение“: алгоритъм *Top-Down (with memoization)*

Режим: Самобучение

Режимът „Самообучение“ е подобен на „Обучение“ с тази разлика, че в този режим обучаемият изпълнява постъпково алгоритъма, като попълва клетките от таблицата. Грешен отговор се индикира чрез оцветяването му в червено, а пояснява на начина на изчисляване очакваната стойност се появява в полето „подсказка“ (*Hint*). Визуалното представяне на ДП-таблицата, както и на кода на алгоритъма с текущо изпълнявания оператор от него, подпомага разбирането на ДП, както и отпадането на необходимостта от използването на „лист и химикал“. Предимство на обучаващото средство е и възможността за задаване потребителски или случайно генериран вход на задачата.

При този режим няма контроли за анимационен и постъпков режим на изпълнение.

Режим: Тест

Този режим се използва при изпитни процедури. Резултатът от теста се пресмята автоматично като общ брой точки (който се визуализира) и се съхранява като цялостно решение във файл. Докладваният инструмент може да се използва и при дистанционно обучение, предоставяйки защита на резултатите от теста чрез използване на криптографска хеш-функция.

Разработеният софтуерен продукт дава възможност на преподавателя за автоматична обработка на резултатите от проведените тестове, както и за директен достъп до резултатите на даден студент като ДП-таблица с маркирани допуснатите грешки.

ЗАКЛЮЧЕНИЕ

Докладът представя специално разработено софтуерно средство за визуализация на една от основните техники за дизайн на алгоритми - „Динамично програмиране“, приложена върху задачата за раницата. Представеният инструмент за обучение подпомага както обучаемия, така и преподавателя в различни етапи от процеса на обучение, самообучение и тестване в условия на присъствен или дистанционен учебен процес. Предимство на представената разработка е визуализацията на двата основни подхода за ДП – итеративния и рекурсивния с мемоизация, както и на алгоритъма за генериране на крайното решение.

Опитът от практическите занятия показва, че разработеното обучително средство увеличава доброто разбиране на концепцията на ДП.

БЛАГОДАРНОСТИ

Този доклад се публикува с подкрепата на проект 24-ФЕЕА-01 „Методи и средства за автоматизиран анализ, обработка, съхранение и управление на мултимедийни данни и документи“, финансиран от фонд „Научни изследвания“ на Русенски университет „Ангел Кънчев“.

REFERENCES

Algorithm Visualizer (2024). Available at: https://algorithm-visualizer.org/dynamic-programming/knapsack-problem.

Alsuwaiyel, M. (2010) *Algorithms: Design Techniques and Analisys*. World Scientific Publishing.

Cormen, T. H. *et al.* (2012) *Introduction to Algorithms*. Third, *Computer*. Third. Edited by PHI Learning. The MIT Press. doi: 10.2307/2583667.

Dasgupta S., C. Papadimitriou and U. Vazirani (2007) *Algorithms*. McGraw-Hill Higher Education.

Dynamic Programming Visualization (2024). Available at: https://dynamic-

programming.debkbanerji.com/problem/knapsack-without-repetition.

Interactive algorithms. Solving big problems by combining small decisions Dynamic programming. Knapsack 01. (2024). Available at: https://hope.simons-rock.edu/~mbarsky/utor/explorables/OWL/issue1_30112015/knapsackexplore.html.

Knapsack Algorithm Visualization (2024a). Available at: https://monicagranbois.com/knapsack-algorithm-visualization/.

Knapsack Algorithm Visualization (2024b). Available at: https://rose-paul.github.io/DP_vizualization/.

Knapsack Visualizer (2024). Available at: https://knapsack.masao.io/.

Levitin, A. (2012) *Introduction to Design and Analysis of Algorithms*. Third. Pearson.

Price, B., Baecker, R. and Small, I. (1998) 'An introduction to software visualization', *Software Visualization*, pp. 3–27.

Ramani, K. V. and Rama Rao, T. P. (1994) 'A graphics based computer-aided learning package for integer programming: The branch and bound algorithm', *Computers & Education*, 23(4), pp. 261–268.

Shaffer, C. A., Cooper, M. and Edwards, S. H. (2007) 'Algorithm visualization: a report on the state of the field', in *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pp. 150–154. doi: 10.1145/1227504.1227366.

Velázquez-Iturbide, J. A. Debdi, O., Esteban-Sánchez, N. and Pizarro, C. (2013) 'GreedEx: A visualization tool for experimentation and discovery learning of greedy algorithms.', *IEEE Transactions on Learning Technologies*, 6(2), pp. 130–143.

Velázquez-Iturbide, J. Á., Pérez-Carrasco, A. and Urquiza-Fuentes, J. (2009) 'A design of automatic visualizations for divide-and conquer algorithms', *Electronic Notes in Theoretical Computer Science 224*, pp. 159–167.

Yi, J. S. *et al.* (2008) 'Understanding and characterizing insights: how do people gain insights using information visualization?', in *2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*, pp. 1–6. doi: 10.1145/1377966.1377971.