AUTOMATYKA • 2011 • Tom 15 • Zeszyt 2

Mariusz Makuchowski*

Algorytm poszukiwania z zabronieniami dla problemu centralnego drzewa rozpinającego

1. Wprowadzenie

W wielu dziedzinach życia poszukujemy rozwiązania problemu, w którym nie wszystkie dane są precyzyjne i z góry znane. W takim przypadku zaczynamy szczególnie cenić rozwiązania odporne na zakłócenia danych wejściowych. Jednym z kryteriów oceny odporności rozwiązania jest odległość od jeszcze nieznanego (na etapie rozwiązywania, a zależnego od czynników losowych) rozwiązania optymalnego. Dlatego często poszukuje się rozwiązania możliwie bliskiego wszystkim innym rozwiązaniom (w szczególności bliskie jeszcze nieznanemu rozwiązaniu optymalnemu) – tak zwanego rozwiązania centralnego.

Koncepcja centralnego drzewa rozpinającego graf pojawiła się po raz pierwszy pracy [1]. W pracy [2] pokazano, że problem znalezienia centralnego drzewa należy do klasy problemów NP-trudnych, a to uzasadnia stosowanie przybliżonych algorytmów heurystycznych. Natomiast z pracy [3] wynika, że dla dowolnego algorytmu mamy gwarancję, iż wartość funkcji celu dostarczonego rozwiązania nie będzie gorsza więcej niż dwukrotnie względem wartości funkcji celu rozwiązania optymalnego. W pracy [4] przedstawiono zachłanny algorytm konstrukcyjny oraz algorytm lokalnych popraw.

2. Problem centralnego drzewa rozpinającego

Przed zdefiniowaniem problemu centralnego drzewa rozpinającego (*Central Spanning Tree Problem*) wprowadzimy najpierw podstawowe oznaczenia, definicje i miary zgodne z notacją zawartą we wstępie pracy [5].

Na wejściu problemu mamy zadany spójny nieskierowany graf $G=(V, E)$ będący parą: zbioru wierzchołków V i zbioru krawędzi E. Analizowany problem polega na znalezieniu pewnego drzewa rozpinającego ten graf. Przestrzenią rozwiązań jest zbiór wszystkich możliwych drzew rozpinających graf G, zbiór ten będzie dalej oznaczany przez $\Gamma(G)$.

* Politechnika Wrocławska, Instytut Informatyki, Automatyki i Robotyki

Znany w teorii jest fakt iż, liczność zbioru $\Gamma(G)$ zależy zarówno od struktury grafu G, jak i od jego rozmiaru. Dla grafu pełnego zachodzi zależność:

$$|\Gamma(G)| = |V|^{|V|-2} \tag{1}$$

Rozmiar przestrzeni rozwiązań może więc rosnąć wykładniczo wraz ze wzrostem ilości danych wejściowych. Poszukiwane drzewo rozpinające ma być centralnym drzewem, a więc takim, które znajduje się możliwie blisko centrum przestrzeni rozwiązań. Oznacza to, że maksymalna odległość od pozostałych drzew powinna być jak najmniejsza, według danej miary odległości. W literaturze [2, 4] miara odległości pomiędzy drzewami $T_1, T_2 \in \Gamma(G)$ rozpiętymi na grafie G przyjmuje postać:

$$Dist(T_1, T_2) = \frac{1}{2}\left|(E(T_1) \cup E(T_2)) \setminus (E(T_1) \cap E(T_2))\right| \tag{2}$$

gdzie $E(T)$ oznacza zbiór krawędzi drzewa $T \in \Gamma(G)$. W literaturze [6, 4] spotyka się także równoważny opis powyższej miary w bardziej przystępnej postaci:

$$Dist(T_1, T_2) = |E(T_1) \setminus E(T_2)| \tag{3}$$

Dla ustalonego drzewa $T \in \Gamma(G)$ można wyznaczyć, w czasie wielomianowym względem danych wejściowych, drzewo $\overline{T} \in \Gamma(G)$, które jest możliwie najdalej położone w przestrzeni rozwiązań w sensie miary opisanej wzorami (2), (3);

$$\overline{T} \in \arg \max_{T' \in \Gamma(G)} Dist(T, T') \tag{4}$$

W celu wyznaczenia drzewa $\overline{T} \in \Gamma(G)$ należy nadać krawędziom drzewa T wagi 1, a pozostałym wagi 0. Następnie należy wyznaczyć minimalnokosztowe drzewo rozpinające według algorytmów podanych w pracach [7, 8]. Łączna suma wag krawędzi tak otrzymanego drzewa jest dokładnie odległością pomiędzy tymi drzewami. Dystans pomiędzy danym drzewem $T \in \Gamma(G)$ i drzewem od niego najbardziej oddalonym $\overline{T} \in \Gamma(G)$ będziemy oznaczać przez:

$$\Delta(T) = Dist(T, \overline{T}) = \max_{T' \in \Gamma(G)} Dist(T, T') \tag{5}$$

Problem centralnego drzewa rozpinającego graf G sprowadza się do znalezienia takiego rozwiązania, dla którego dystans do drzewa najdalej od niego położonego jest możliwie najmniejszy:

$$T^* \in \arg \min_{T \in \Gamma(G)} \Delta(T) = \arg \min_{T \in \Gamma(G)} \max_{T' \in \Gamma(G)} |E(T) \setminus E(T')| \tag{6}$$

3. Algorytm tabu

Prezentowany w pracy algorytm, oparty na ogólnej idei poszukiwania z zabronieniami, będziemy dalej oznaczać CSTTS (*Central Spanning Tree Taboo Search*). Bazuje on na idei przedstawionej w pracach [9, 10]. Poszczególne elementy bazowe proponowanego algorytmu opisane są szczegółowo w poniższych rozdziałach.

3.1. Sąsiedztwo

Zmienną decyzyjną w rozpatrywanym problemie jest drzewo T rozpięte na spójnym nieskierowanym grafie G. W wyniku przeszukiwania przestrzeni rozwiązań chcemy przeglądać drzewa leżące możliwie blisko bieżącego rozwiązania w sensie pewnej miary. Kryterium oceny rozwiązania stosuje miarę (2), (3) odległości pomiędzy drzewami, dlatego jako sąsiedztwo $N(T)$ bieżącego rozwiązania T przyjmujemy zbiór wszystkich drzew rozpinających graf G, dla których odległość (w sensie tych miar) wynosi dokładnie jeden:

$$N(T) = \{T' \in \Gamma(G) : |E(T) \setminus E(T')| = 1\} \tag{7}$$

Liczność sąsiedztwa $N(T)$ może być ograniczona od góry przez wielomian:

$$|N(T)| = O(|E||V|) \tag{8}$$

W badaniach wstępnych analizowane były również inne rodzaje sąsiedztw w szczególności będące podzbiorami sąsiedztwa (7). Z analizy wstępnych rezultatów wynika, że algorytm wyposażony w sąsiedztwo prezentowane w niniejszej pracy był algorytmem najszybciej zbieżnym, tzn. najszybciej osiągający optymalne rozwiązania. Tylko ten algorytm jest prezentowany w bieżącej pracy.

3.2. Lista tabu

Z definicji sąsiedztwa wynika, że pojedynczy ruch oznaczany (e, s) w algorytmie CSTTS, transformujący drzewo $T1$ w drzewo $T2 \in N(T1)$, polega na usunięciu pewnej krawędzi $e \in E(T1)$ z drzewa $T1$ i zastąpieniu jej krawędzią $s \in E \setminus E(T1)$ taką, że $(V, E(T1) \setminus \{e\} \cup \{s\}) \in \Gamma(G)$. W algorytmie są dwie listy tabu, na pierwszej z nich *eLT* zapisywane są usuwane krawędzie, a na drugiej z nich *sLT* krawędzie dodawane. Obydwie listy mają skończoną długość i pamiętają odpowiednio *eLTN* i *sTLN* ostatnio dodanych do nich elementów. Ruch (e, s) uważany jest za tabu, jeżeli usuwana krawędź e znajduje się na liście *sLT* lub jeżeli dodawana krawędź s znajduje się na liście *eLT*.

3.3. Rozwiązanie startowe i dodatkowe techniki dywersyfikacji

Rozwiązanie startowe algorytmu jest pseudolosowym drzewem rozpinającym graf G. W algorytmie zastosowano także metodę dywersyfikacji poszukiwań rozwiązań. Polega ona na restarcie algorytmu w przypadku stagnacji obliczeń. Jeżeli algorytm przez określoną

liczbę iteracji nie poprawił rozwiązania, to następuje zastąpienie bieżącego rozwiązania pseudolosowym drzewem. Liczba jałowych iteracji, po której następuje opisany mechanizm dywersyfikacji, nazywana jest *restart.*

4. Badania numeryczne

Wszystkie prezentowane wyniki badań w szczególności czasy pracy algorytmów uzyskane były na komputerze klasy PC wyposażonym w procesor Core2Duo E6750 2,66 GHz, 4 GB pamięci RAM, pracującym w systemie Windows 7. Algorytmy zaprogramowane zostały jako jednowątkowe w środowisku DEV C++ 4.9.9.1.

Badania numeryczne przeprowadzone zostały na wyjątkowo trudnych 100 przykładach. Przykłady te stworzone zostały na podstawie idei zaproponowanej w pracy [11]. Przykłady podzielone są na 10 grup oznaczanych rzymskimi liczbami I–X. W ramach danej grupy grafy posiadają jednakową liczbę wierzchołków oraz krawędzi. Liczba wierzchołków zmienia się linowo od 10 do 100, a liczba krawędzi zmienia się od 24 do 1374. Przykłady te charakteryzują się dużą liczbą szerokich minimów lokalnych oraz jednym minimum globalnym. Ponadto ze względu na ich charakterystyczną budowę znane są rozwiązania optymalne, co ułatwia ocenę efektywności badanego algorytmu. Instancje przykładów w ramach pojedynczej grupy opisują de facto ten sam graf, lecz inaczej są etykietowane wierzchołki i w innej kolejności zapisane są kolejne węzły i krawędzie w danych wejściowych grafu.

W celu praktycznej oceny trudności testowanych przykładów oraz stworzenia punktu odniesienia dla czasu pracy algorytm CSTTS został stworzony model programowania liniowego, a następnie rozwiązany przy pomocy pakietu IBM ILOG OPL v6.3. Czas pracy pakietu dla poszczególnych grup przedstawiony został w tabeli 1.

Tabela 1

Czas pracy pakietu ILOG OPL dla poszczególnych grup instancji

Grupa	Rozmiar (|V|, |E|)	Minimalny czas [s]	Średni czas [s]	Maksymalny czas [s]
I	(10, 24)	0,09	0,10	0,12
II	(20, 74)	0,21	0,35	0,68
III	(30, 149)	0,73	1,56	2,83
IV	(40, 249)	2,54	8,46	15,78
V	(50, 374)	13,72	44,77	130,21
VI	(60, 524)	55,78	227,96	572,05
VII	(70, 699)	408,69	1231,55	3965,62

Następnie dla każdego z przykładów uruchomiony został algorytm CSTTS. Parametry, przy których proponowany algorytm osiągną rozwiązania optymalne, przy możliwie najkrótszym czasie działania, to: *liczba iteracji* = 1000, *restart* = 400, *eLT* = 0, *sLT* = 10. W tabeli 2 przedstawiony został średni czas pracy algorytmu CSTTS, oraz średni czas, po jakim algorytm osiągał rozwiązanie optymalne.

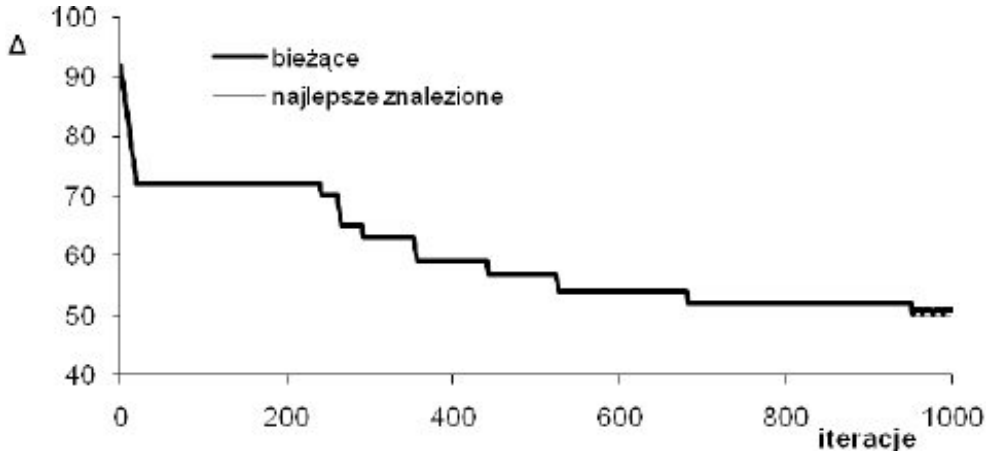
Tabela 2

Czas pracy algorytmu CSTTS (*liczba iteracji* = 1000, *restart* = 400, *eLT* = 0, *sLT* = 10) dla poszczególnych grup instancji

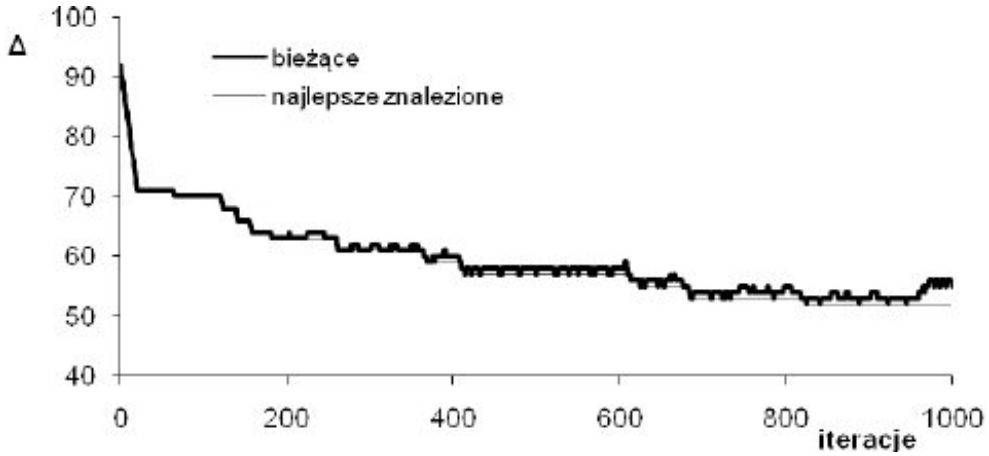
Grupa	Rozmiar (|V|, |E|)	Średni czas osiągnięcia opt. [s]	Średni czas pracy [s]
I	(10, 24)	0,01	0,03
II	(20, 74)	0,01	0,26
III	(30, 149)	0,09	0,93
IV	(40, 249)	0,25	2,27
V	(50, 374)	1,04	4,87
VI	(60, 524)	2,03	9,21
VII	(70, 699)	3,08	15,16
VIII	(80, 899)	9,09	25,09
IX	(90, 1124)	14,56	36,18
X	(100,1374)	32,90	56,40

Przykładowe dwa przebiegi dla tego samego przykładu z X grupy przedstawione są na rysunkach 1 i 2. Rysunek 1 przedstawia przebieg algorytmu z doświadczalnie dobraną możliwie najlepszą długością list tabu. Długość listy tabu jest na tyle duża, by algorytm nie wpadał w cykle i jednocześnie na tyle mała, by zbyt szybko nie opuszczał szerokich płaskich minimów w przestrzeni rozwiązań. Nie jest to typowy przebieg algorytmu opartego na technice poszukiwania z zabronieniami. Kształt owego przebiegu zdeterminowany jest przez nietypowy i zarazem bardzo trudny krajobraz przestrzeni rozwiązań. Dlatego na rysunku 2 zaprezentowany został także przebieg algorytmu dla większych długości listy tabu. Zwiększenie długości listy tabu spowodowało, iż charakter zmiany wartości funkcji celu wzorcowo odpowiada idei poszukiwania z zabronieniami. Zwiększenie długości list tabu powodowało jednak wyraźny spadek wydajności algorytmu, mierzony jako średni błąd od rozwiązań optymalnych w sensie różnicy wartości funkcji celu. Podobnie stosowanie dwóch list tabu (jednej zabraniającej usuwania, a drugiej dodawania krawędzi do drzewa) powodowało spadek efektywności algorytmu. Ostatecznie najlepszą strategią tabu okazała

się technika zabronień dodawania krawędzi, które były usunięte w ostatnich 10 iteracjach pracy algorytmu, przy jednoczesnym całkowitym wyłączeniu zabronień usuwania krawędzi (*eLTN* = 0, *sLTN* = 10).



Rys. 1. Przebieg algorytmu CSTTS dla jednego przykładu z 10 grupy, wysterowanie algorytmu: *liczba iteracji* = 1000, *restart* = 400, *eLT* = 0, *sLT* = 10



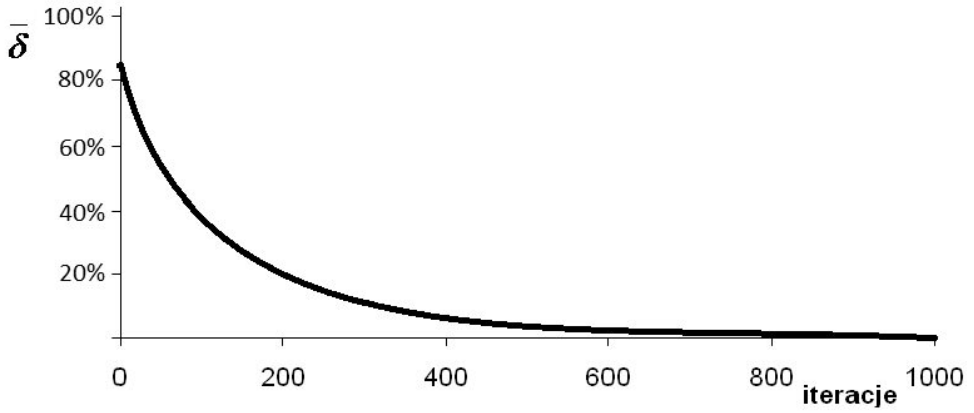
Rys. 2. Przebieg algorytmu CSTTS dla jednego przykładu z 10 grupy, wysterowanie algorytmu: *liczba iteracji* = 1000, *restart* = 400, *eLT* = 20, *sLT* = 0

Oczywiście jak w każdym algorytmie popraw czas pracy algorytmu można skracać kosztem jakości uzyskiwanych rozwiązań. Na rysunku 3 przedstawiona została zależność średniej jakości uzyskiwanych rozwiązań względem ilości wykonanych przez algorytm ite-

racji. Przy ustalonej liczbie iteracji, dla każdego z przykładów policzony został δ względny błąd uzyskanego rozwiązania $\Delta(T)$ zgodnie z wzorem:

$$\delta = 100\% \cdot \frac{\Delta(T) - \Delta(T^*)}{\Delta(T^*)} \tag{9}$$

a następnie na wykresie zaznaczono $\bar{\delta}$ jego średnią wartość z wszystkich 100 instancji testowych.



Rys. 3. Zależność średniego błędu rozwiązań algorytmu CSTTS w zależności od wykonanych iteracji dla ustawień *restart* = 400, *eLT* = 0, *sLT* = 10

W przykładach testowych, niezależnie od wielkości instancji, średni błąd losowych rozwiązań utrzymywał się na poziomie 85%. Zgodnie z intuicją największa poprawa rozwiązań następowała w początkowej fazie pracy algorytmu. Dla 1000 iteracji algorytm osiąga optymalne rozwiązania we wszystkich 100 testowanych instancjach.

5. Podsumowanie

Zaproponowany w pracy algorytm CSTTS bazujący na schemacie poszukiwania z zabronieniami wykazuje się bardzo dużą skutecznością działania. Dla prezentowanych przykładów osiągał on zawsze optymalne rozwiązanie w czasie krótszym niż jedna minuta. Dla innych przykładów, nieprezentowanych w pracy, które tylko udało się rozwiązać optymalnie przez pakiet ILOG OPL v6.3 w ciągu kilkudziesięciu godzin, algorytm CSTTS również znajdował optimum w co najwyżej kilkadziesiąt sekund.

Prezentowany algorytm łatwo można zmodyfikować dla bardziej ogólnego problemu, jakim jest znajdowanie minimalnokosztowego odpornego drzewa rozpinającego z przedziałowymi obciążeniami krawędzi (*min-max regret minimum spanning tree with interval*

data). Wstępne badania wykazują, że jest on wtedy bardziej wydajny niż inne literaturowe algorytmy dedykowane temu problemowi.

Badania finansowane przez Ministerstwo Nauki i Szkolnictwa Wyższego, grant nr N N206 492938.

Literatura

[1] Deo N., *A central tree.* IEEE Transactions on Circuit Theory, vol. ct-13, 1966, 439–440.
[2] Bezrukov S., Kaderali F., Poguntke W., *On central spanning trees of a graph.* Lecture Notes Computer Science, 1120, 1996, 53–58.
[3] Kasperski A., Zieliński P., *An approximation algorrithm for interwal data minmax regret combinatorial optimalization problems.* Information Processing Letters, 97(5), 2006, 177–18.
[4] Bang-Jensen J., Nikulin Y., *Heuristics for the central tree problem.* Journal of Heuristics, 16, 2010, 663–651.
[5] Harary F., *Graph theory.* Addison-Wesley Publ. Company, 1969.
[6] Aron I., Hentenryck P., *On the complexity of the robust spanning tree problem with interval data.* Operations Research Letters, 32, 2004, 36–40.
[7] Kruskal J.B., *On the shortest spanning subtree of graph and the trveling salesman problem..* KProc. Amer. Math. Soc., 7, 1956, 48–50.
[8] Prim R.C., *Shortest connection networks and some generalizations.* Bell System Technical Journal, 36, 1957, 1389–1401.
[9] Glover F., Laguna M., *Tabu Search..*Kluwer Academic Publishers, Massachusetts, USA, 1997.
[10] Nowicki E., *Metoda tabu w problemach szeregowania zadań produkcyjnych.* Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1999.
[11] Kasperski A., *Discrete optimalization with interval data: Minmax regret and fuzzy approach.*Studies in Fuzziness and Soft Computing, vol 228. Springer-Verlag, Berlin Heidelberg 2008.