Radosław GRYMIN, Wojciech BOŻEJKO, Jarosław PEMPERA,
Mieczysław WODECKI
Politechnika Wrocławska

ALGORYTM ROZWIĄZYWANIA DYSKRETNO-CIĄGŁEGO PROBLEMU INSPEKCJI

Streszczenie. W pracy definiujemy nowy problem inspekcji będący uogólnieniem klasycznego problemu komiwojażera, a polegający na odwiedzeniu ciągłych obszarów (kół) w taki sposób, by zminimalizować przebytą drogę. Problem pojawia się w praktyce jako zagadnienie marszrutyzacji bezzałogowego statku powietrznego i ma charakter dyskretno-ciągły. Do jego rozwiązywania proponujemy zastosowanie algorytmów lokalnych poszukiwań.

THE ALGORITHM FOR SOLVING A DISCRETE-CONTINUOUS INSPECTION PROBLEM

Summary. In the article we define the new inspection problem that is a generalization of the classical Travelling Salesman Problem. Its idea is to visit continuous areas (circles) in a way, that minimizes travelled distance. The problem appears in practice as a problem of scheduling unmanned aerial vehicle and has dicrete-contnuous nature. In order to solve it we propose use of local search algorithms.

1. Wprowadzenie

W ostatnich latach, bezzałogowe statki powietrzne (ang. *unmanned aerial vehicle*, UAV) stały się bardzo popularne w wielu dziedzinach. Poza profesjonalnym i amatorskim nagrywaniem filmów, stosowane są do inspekcji. Dużym problemem jest krótki czas lotu w porównaniu do czasu wymaganego do pełnego naładowania baterii. Na przykład, jeden z najnowocześniejszych bezzałogowych statków powietrznych DJI Matrice 200 wykorzystywany w inspekcji może latać od 13 do 38 minut aż do całkowitego rozładowania (w zależności od baterii, wyposażenia i prędkości przelotu), a jego czas ładowania wynosi 2 godziny 30 minut. Dlatego też autonomiczna inspekcja z wykorzystaniem bezzałogowego statku latającego wymaga stosowania efektywnych algorytmów wyznaczających najkrótszą trasę lotu.

2. Problem Inspekcji

Danych jest n obiektów, dla których należy przeprowadzić inspekcję $\{O_1, O_2, O_3, \ldots, O_n\}$. Dla każdego obiektu $O_i, i \in \{1, 2, 3, \ldots, n\}$ istnieje dopuszczalna, trójwymiarowa podprzestrzeń (obszar) A_i, z której mogą zostać wykonane czytelne

zdjęcia. Rozważany uogólniony problem optymalizacyjny polega na znalezieniu punktów najkrótszej ścieżki, która odwiedza wszystkie obszary co najmniej raz (aby wykonać zdjęcia) i kolejność, w której te punkty powinny zostać odwiedzone, a następnie powrócić do punktu początkowego.

Taki problem stanowi uogólnienie (ciągłą wersję) problemu *Set TSP* (*GTSP*), znanego również jako: uogólniony (*generalized*) *TSP*, *International TSP*, *Group TSP, one-of-a-set TSP, coverng salesman problem*, *multiple choice TSP*. Problem jest silnie NP-trudny ponieważ może zostać zredukowany do *TSP* jeżeli obszary są pojedynczymi punktami.

GTSP pojawia się często w problemach planowania ruchu (Imeson i Smith [5]; Mathew et al. [7]; Wolff et al. [11]). Dla tej klasy problemów zostało zaproponowanych w literaturze wiele heurystyk i algorytmów dokładnych, m.in. algorytmy ewolucyjne [10], również w wersji równoległej [2]. Zgodnie z naszą najlepszą wiedzą brak jest w literaturze prac, w których rozważa się dyskretno-ciągłą wersję problemu *GTSP*.

W dalszych rozważaniach będziemy zakładać, że przestrzeń widoczności obiektu poddawanego inspekcji może być aproksymowany za pomocą półkuli (np. w przypadku fotografowania obiektów oraz komunikacji nadajnik-odbiornik radiowy) i że bezzałogowy statek powietrzny lata na stałym pułapie. Wtedy obszar inspekcji dla każdego obiektu inspekcji może zostać opisany za pomocą koła. To koło będzie nazywane przez nas *kołem widoczności*, natomiast punkt, w którym wykonana jest inspekcja (np. fotografowanie) będzie nazywane *punktem inspekcji*.

2.1. Uproszczony problem inspekcji

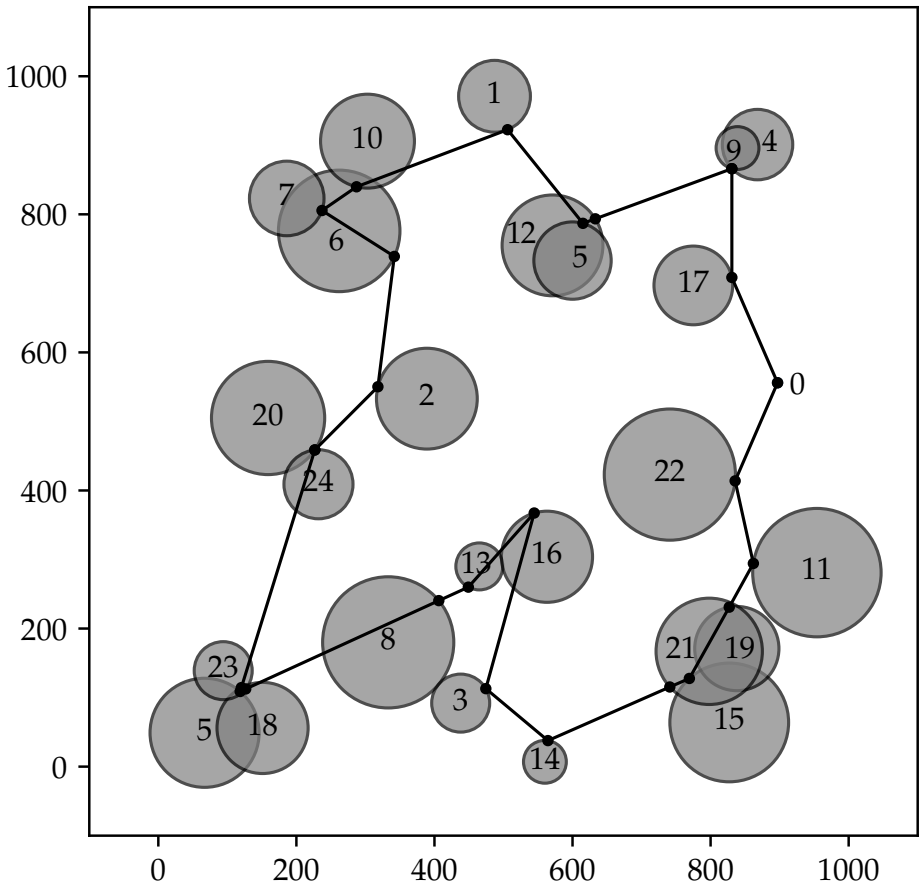
Danych jest n obiektów ze zbioru $N = 1, \ldots, n$, dla których należy przeprowadzić inspekcję za pomocą bezzałogowego statku powietrznego. Dla każdego obiektu $i \in N$, dana jest przestrzeń widoczności określona za pomocą trójki (X_i, Y_i, R_i), $R_i > 0$, gdzie X_i oraz Y_i oznaczają współrzędne środka, natomiast R_i oznacza promień koła widzialności obiektu. Przed rozpoczęciem lotu, bezzałogowy statek powietrzny znajduje się w punkcie o współrzędnych (x_0, y_0). Po skończonej inspekcji bezzałogowy statek powietrzny musi wrócić do tego punktu. Celem optymalizacji jest określenie punktu inspekcji dla każdego obiektu oraz znalezienie kolejności odwiedzania punktów inspekcji takiej, dla której długość przebytej drogi przez bezzałogowy statek powietrzny jest najkrótsza. Dla zobrazowania problemu przykładowe rozwiązanie dopuszczalne (suboptymalne) pewnej instancji problemu jest zaprezentowane na Rysunku 1.

Niech $r = \{0, r_1, r_2, \ldots, r_n, 0\}, r_i \in N$ oznacza kolejność inspekcji obiektów (zero oznacza punkt początkowy). Niech x_i oraz y_i będą współrzędnymi punktu inspekcji dla koła widoczności obiektu i. Punkt (x_i, y_i) musi należeć do koła określonego za pomocą trójki (X_i, Y_i, R_i). Dla danej kolejności inspekcji r i danych punktów inspekcji (x_i, y_i), $i \in N$ długośc drogi jest równa

$$L(r, x, y) = \sum_{s=1}^{n+1} d(x_{r_{s-1}}, y_{r_{s-1}}, x_{r_s}, y_{r_s}), \tag{1}$$

gdzie $x = (x_0, x_1, \ldots, x_n)$, $y = (y_0, y_1, \ldots, y_n)$, natomiast $d(x_k, y_k, x_l, y_l)$ jest odległością euklidesową pomiędzy punktami (x_k, y_k) i (x_l, y_l). Rozważany problem optymalizacyjny może zostać podzielony na dwa poziomy:

- niski poziom – określenie optymalnych punktów inspekcji dla zadanej sekwencji r,



Rys. 1. Ilustracja kół widoczności, punktu początkowego 0 (zero) oraz ścieżki znalezionej przez algorytm

- wysoki poziom – określenie optymalnej sekwencji r.

Każdy z poziomów wymaga użycia innego rodzaju algorytmu. Niski poziom wymaga użycia algorytmów optymalizacji ciągłej, a wysoki poziom – algorytmu optymalizacji dyskretnej. W dalszej części pracy opisane zostaną zaproponowane algorytmy użyte na niskim i wysokim poziomie rozwiązania problemu.

2.2. Procedura niskiego poziomu wyznaczania punktów w obszarach

Na wstępie należy zwrócić uwagę, że niezależnie od wyboru punktów inspekcji w kole widoczności, ścieżka lotu bezzałogowego statku powietrznego ma co najmniej jeden punkt wspólny z obwodem koła. Weźmy jeden dowolnie arbitralnie wybrany punkt wspólny ścieżki bezzałogowego statku powietrznego i obwodu koła widoczności. Dla obiektu $i \in N$ ten punkt może być jednoznacznie określony za pomocą kąta α_i. Współrzędne kartezjańskie są określone za pomocą równań:

$$x_i(\alpha_i) = X_i + R_i \cos(\alpha_i), \tag{2}$$

$$y_i(\alpha_i) = Y_i + R_i \sin(\alpha_i). \tag{3}$$

Dla $\alpha = (\alpha_1, \ldots, \alpha_n)$ długość ścieżki jest równa

$$\begin{array}{rcl} L(r,\alpha) & = & d(x_0, y_0, x_{r_1}(\alpha_{r_1}), y_{r_1}(\alpha_{r_1})) \\ & + & \sum_{s=1}^{n-1} d(x_{r_s}(\alpha_{r_s}), y_{r_s}(\alpha_{r_s}), x_{r_{s+1}}(\alpha_{r_{s+1}}), y_{r_{s+1}}(\alpha_{r_{s+1}})) \\ & + & d(x_{r_n}(\alpha_{r_n}), y_{r_n}(\alpha_{r_n}), x_0, y_0). \end{array} \tag{4}$$

Dla danej sekwencji r, $L(r,\alpha)$ jest nieliniową funkcją o n zmiennych. W celu określenia minimalnej wartości funkcji $L(r,\alpha)$ może zostać użytych wiele rodzajów algorytmów optymalizacyjnych , np. algorytm Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) (Byrd, Lu i Nocedal [3], Zhu et al. [12]), metoda Nelder-Mead (Nelder i Mead [8]), czy metoda SLSQP (Sequential Least Squares Programming, Kraft [6]). Zwykle użycie efektywnych algorytmów optymalizacji ciągłej w celu rozwiązania problemu niskiego poziomu wiąże się z długim czasem obliczeń.

Załóżmy, że bezzałogowy statek powietrzny porusza się z punktu o współrzędnych (x_a, y_a) przez punkt inspekcji obiektu b do punktu o współrzędnych (x_c, y_c). Kąt α_b jest optymalnym kątem $(x_b(\alpha_b), y_b(\alpha_b))$ obiektu b jeśli $d(x_a, y_a, x_b(\alpha_b), y_b(\alpha_b)) + d(x_b(\alpha_b), y_b(\alpha_b), x_c, y_c)$ ma minimalną wartość. Punkt wyznaczony w ten sposób będzie nazywany lokalnie optymalnym punktem. Ze względu na ciągły charakter optymalizowanej funkcji, w celu określenia kąta α_b można wykorzystać szybką metodę Powell'a (Powell [9]).

Początkowymi parametrami algorytmu są początkowa trasa oraz początkowy ciąg kątów $\alpha^0 = (\alpha_1^0, \ldots, \alpha_n^0)$. W każdej iteracji proponowanego algorytmu dla danego $\alpha = (\alpha_1, \ldots, \alpha_n)$ i dla każdego obiektu $s \in N$ określony jest lokalnie optymalny kąt α_s' oraz zmiana długości trasy opisana za pomocą równania:

$$\begin{array}{rcl} \Delta(s, \alpha_s, \alpha_s') & = & d(x_{\bar{s}}, y_{\bar{s}}, x_s(\alpha_s), y_s(\alpha_s)) + d(x_s(\alpha_s), y_s(\alpha_s), x_{\underline{s}}, y_{\underline{s}}) \\ & - & d(x_{\bar{s}}, y_{\bar{s}}, x_s(\alpha_s'), y_s(\alpha_s')) - d(x_s(\alpha_s'), y_s(\alpha_s'), x_{\underline{s}}, y_{\underline{s}}), \end{array} \tag{5}$$

gdzie $\bar{s}$ jest bezpośrednim poprzednikiem, natomiast $\underline{s}$ jest bezpośrednim następnikiem obiektu s w sekwencji r. Oczywiście, poprzednikiem pierwszego obiektu w kolejności r jest punkt początkowy, tj. $r(1) = 0$, następnikiem ostatniego obiektu w trasie jest również ten punkt, tj, $r(n) = 0$.

Algorytm generuje ciąg α składający się z punktów inspekcji. Zatrzymuje swoje wykonanie jeśli w kolejnych iteracjach długość drogi nie skraca się więcej niż arbitralnie wybrana wartość ϵ lub po wykonaniu $MaxIter$ iteracji. Dokładny opis proponowanej metody przedstawia Algorytm 1.

2.3. Procedura wysokiego poziomu wyznaczająca kolejność zadań

Dysponując metodą dolnopoziomową wyznaczania optymalnej trasy (i jej długości $L(r)$) dla danej sekwencji r odwiedzania obszarów inspekcji, problem górnego poziomu sprowadza się do rozwiązania problemu komiwojażera (*TSP*). W rozważanym problemie, pozycja punktów inspekcji zależy od sekwencji odwiedzania obszarów inspekcji, natomiast w *TSP* odległości pomiędzy miastami są stałe. Ta istotna różnica uniemożliwia stosowanie ogólnie znanych technik szybkiego wyznaczenia zmian długości trasy przy małych zmianach sekwencji miast trasy.

W celu wyznaczenia kolejności odwiedzania obszarów inspekcji wykorzystano algorytm 2-Opt oraz zaprojektowano algorytm oparty na metodzie symulowanego wyżarzania.

Algorytm 1: Niskopoziomowy algorytm optymalizacyjny

$\alpha = \alpha^0$
for $iter = 1$ to $MaxIter$ **do**
  for $s = 1$ to n **do**
    determine α'_s and $\Delta(s, \alpha_s, \alpha'_s)$
  end for
  determine such k that $\Delta(k, \alpha_k, \alpha'_k) = \max_{s \in N} \Delta(s, \alpha_s, \alpha'_s)$
  if $\Delta(k, \alpha_k, \alpha'_k) < \epsilon$ **then**
    return α
  end if
  $\alpha_k = \alpha'_k$
end for

Algorytm 2-Opt

Algorytm 2-Opt (Croes [4]) jest najprostszą heurystyką przeszukiwania lokalnego dla problemu komiwojażera. Jest algorytmem zstępującym opartym na otoczeniu wykorzystującym odwrócenie kolejności odwiedzania punktów fragmentu trasy (otoczenie 2-Opt). W każdej iteracji algorytmu dla trasy bieżącej wyznaczane są trasy sąsiednie, z których wybierana jest najlepsza. Algorytm kończy działanie gdy w otoczeniu 2-Opt nie ma rozwiązań lepszych od bieżącego.

Trasa sąsiednia trasy r konstruowana jest w oparciu o modyfikację opisaną przy pomocy pary (a, b). Nowa trasa przyjmuje postać

$$r(a, b) = (r_1, \ldots, r_{a-1}, r_b, r_{b-1}, \ldots, r_{a+1}, r_a, r_{b+1}, \ldots, r_n). \tag{6}$$

Wygenerowane w ten sposób wszystkie rozwiązania tworzą otoczenie trasy r.

Algorytm 2-Opt nie gwarantuje znalezienia optymalnej drogi, ale za to zapewnia (w problemie euklidesowym), że wynikowa ścieżka nie będzie przecinać się sama ze sobą. Jego dużą zaletą jest prostota i zazwyczaj szybki czas wykonania.

Algorytm Symulowanego Wyżarzania

Algorytm Symulowanego Wyżarzania (*simulated annealing, SA*) nawiązuje do termodynamicznego procesu wyżarzania metalu, gdzie próbka metalu jest poddawana procesowi studzenia w celu osiągnięcia pożądanych właściwości (twardości, elastyczności, itp.). Idea działania *SA* sprowadza się do wygenerowania trajektorii poszukiwań (gdzie każde kolejne rozwiązanie jest wybierane losowo z otoczenia bieżącego rozwiązania) opartej na losowych ruchach w otoczeniu. Wylosowane rozwiązanie jest akceptowane (zastępuje rozwiązanie bieżące w kolejnej iteracji) bezwarunkowo, gdy jest niegorsze od bieżącego. Możliwe jest również zaakceptowanie rozwiązań gorszych z prawdopodobieństwem zależnym np. od temperatury i różnicy wartości funkcji celu (tzw. funkcja akceptacji). Szczegółowy opis metody można znaleźć w pracy [1]. W algorytmie zaimplementowanym do rozwiązywania rozważanego problemu zastosowano otoczenie 2-Opt. Algorytm wykonywał $MaxIter = 2000$ iteracji. W rezultacie, trajektoria poszukiwań algorytmu SA jest prowadzona w „statystycznie dobrym" kierunku.

Ostatecznie zostały zaimplementowane trzy algorytmy: 2-OPT(C), 2-OPT oraz SA. W algorytmie 2-OPT(C) długość trasy wyznaczono w oparciu o lokalizacje obiek-

tów inspekcji tj.

$$LC(r) = d(x_0, y_0, X_{r_1}, Y_{r_1}) + \sum_{s=1}^{n-1} d(X_{r_s}, Y_{r_s}, X_{r_{s+1}}, Y_{r_{s+1}}) + d(X_{r_n}, Y_{r_n}, x_0, y_0) \quad (7)$$

3. Badania eksperymentalne

Wygenerowano 6 grup instancji z identyczną liczbą obiektów do odwiedzenia, $n \in \{5, 10, 15, 20, 25, 30\}$. Każda grupa zawiera 10 instancji problemów wygenerowanych losowo. W każdej instancji, środki kół zostały rozłożone losowo na kwadracie o boku długości 1000. Promienie zostały wygenerowane z rozkładem jednostajnym $U(a, b)$, z wartościami parametrów charakterystycznych dla każdej grupy, $(a, b) \in \{(100, 200), (50, 150), (40, 120), (30, 100), (30, 100), (30, 100)\}$. Instancje o większym rozmiarze musiały zawierać mniejsze kółka, by uzyskać część kółek nachodzących na siebie, a część nienachodzących.

Metaheurystyki 2-OPT(C), 2-OPT i SA były oprogramowane w języku C++ w środowisku Visual Studio 10 i uruchomione na pojedynczym rdzeniu procesora Intel Core i7 3,5 GHz, z 8 GB RAM pod systemem operacyjnym Windows 7. Algorytm SA został zaprojektowany w wersji z samostrojeniem parametrów początkowych (zobacz [1]).

Użyto dwóch miar w celu określenia cech algorytmu. Pierwszą jest czas wykonania (w sekundach), druga odnosi się do jakości algorytmu. W celu porównania efektywności poszczególnych etapów optymalizacji badano względną różnicę (*percentage relative deviation, PRD*) pomiędzy długością drogi r^A zwróconej przez badany algorytm $A \in \{$2-OPT(C), 2-OPT, SA$\}$ i długość najlepszej znalezionej drogi L^*. Miara jest zdefiniowana za pomocą równania:

$$PRD(A) = \frac{L(r^A) - L^*}{L^*} \cdot 100\%. \quad (8)$$

W celu znalezienia najkrótszej ścieżki przelotu bezzałogowego statku powietrznego, stworzyliśmy oprogramowanie, w którym algorytmy uruchamiane były w następującej kolejności:

$$\text{2-OPT(C)} \Rightarrow \text{2-OPT} \Rightarrow \text{SA-1} \Rightarrow \text{SA-2} \Rightarrow \text{SA-3}. \quad (9)$$

Algorytm 2-OPT(C) generuje w bardzo krótkim czasie ścieżkę, która jest poprawiona przez algorytm 2-OPT i trzykrotnie poprzez algorytm SA. Po skończeniu każdej fazy obliczeń, długość najkrótszej drogi oraz czas obliczeń są zachowane. Podczas kolejnej fazy obliczeń, najlepsza ścieżka z poprzedniej fazy staje się początkową ścieżką dla kolejnej fazy.

W Tabeli 1 zaprezentowano średni błąd względny PRD rozwiązań zwracanych przez kolejne fazy algorytmu. Można zauważyć, że długość drogi w przypadku, gdy statek powietrzny podlatuje do obiektu jest znacznie dłuższa niż długość drogi zwróconej za pomocą proponowanej metody. Jest ona dłuższa średnio o 39,14%, natomiast wartość średnia waha się pomiędzy 32,56%, a 46,03%. Te spostrzeżenia potwierdzają dużą skuteczność proponowanych metod wyznaczania trasy dla statku powietrznego.

Warto zauważyć, że zastosowanie proponowanej metody do sekwencji generowanej przez 2-OPT(C) (patrz kolumna 2-OPT(C) $L(r)$) znacząco redukuje długość trasy.

Tabela 1

Średni względny błąd algorytmów liczony względem SA-3 $L(r)$

n	2-OPT(C) $LC(r)$	2-OPT(C) $L(r)$	2-OPT $L(r)$	SA-1 $L(r)$	SA-2 $L(r)$
5	46,03	1,35	0,00	0,00	0,00
10	37,74	4,09	0,55	0,00	0,00
15	32,56	1,79	0,65	0,43	0,00
20	33,25	4,12	3,15	0,10	0,00
25	43,57	6,71	3,29	1,99	0,03
30	41,69	7,08	3,53	0,96	0,32
Średnio	39,14	4,19	1,86	0,58	0,06

Średnia wartość PRD wynosi tylko 4,19% i waha się w graniach od 1,35% do 7,08%. Niestety jego efektywność maleje wraz ze wzrostem liczby obiektów.

W następnych fazach algorytmy operują na trasach wyznaczonych metodą na dolnym poziomie. W przypadku algorytmu 2-OPT średni błąd wynosi 1,86% i waha się od 0% do 3,53%. Błąd ten rośnie wraz ze wzrostem liczby obiektów.

4. Podsumowanie

W pracy rozpatrujemy dyskretno-ciągły problem marszrutyzacji będący uogólnieniem problemu *International TSP* a dotyczący wyznaczania trasy bezzałogowego statku powietrznego. Proponowana przez nas kaskadowa metoda optymalizacyjna pozwoliła na otrzymanie znaczącej poprawy wyników w stosunku do przypadku, w którym statek powietrzny podlatywał bezpośrednio do fotografowanego obiektu, tj. do centrum docelowego obszaru inspekcji.

LITERATURA

1. Aarts E.H.L., van Laarhoven P.J.M. Simulated Annealing: A Pedestrian Review of the Theory and Some Applications. In: Devijver P.A., Kittler J. (eds) Pattern Recognition Theory and Applications. NATO ASI Series (Series F: Computer and Systems Sciences) 30 (1987). Springer, Berlin, Heidelberg.
2. Bożejko W., Wodecki M., Parallel Evolutionary Algorithm for the Traveling Salesman Problem. Journal of Numerical Analysis, Industrial and Applied Mathematics 2(3-4) (2007), 129–137.
3. Byrd R.H., Lu P., Nocedal J., A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing 16(5) (1995), 1190–1208.
4. Croes G. A., A method for solving traveling-salesman problems. Operations Research 6(6) (1958), 791–812.
5. Imeson F., Smith S.L., Multi-robot task planning and sequencing using the SAT-TSP language IEEE International Conference on Robotics and Automation (2015), 5397–5402.
6. Kraft D., A software package for sequential quadratic programming. Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt (1988).

7. Mathew N., Smith S.L., Waslander S.L., Multirobot rendezvous planning for recharging in persistent tasks. IEEE Trans. Robot., 31 (1) (2015), 128–142.
8. Nelder J.A., Mead R., A simplex method for function minimization. The computer journal 7(4) (1965), 308–313.
9. Powell M. J. D., An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal 7(2) (1964), 155—162.
10. Snyder L., Daskin M., A random-key genetic algorithm for the generalized traveling salesman problem. European Journal of Operational Research 17 (1) (2006), 38–53.
11. Wolff E.M., Topcu U., Murray R.M., Optimal control of non-deterministic systems for a computationally efficient fragment of temporal logic. IEEE Conference on Decision and Control (2013), 3197–3204.
12. Zhu C., Byrd RH., Lu P., Nocedal J., Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on Mathematical Software (TOMS) 23(4) (1997), 550–560.