Studia i Materiały Informatyki Stosowanej, Tom 1, Nr 1, 2009

SYSTEMY MRÓWKOWE W ZASTOSOWANIU DO ROZWIĄZANIA PROBLEMU KOMIWOJAŻERA

Mirosław Modrzejewski

Uniwersytet Kazimierza Wielkiego
Instytut Techniki
II rok MU Edukacja Techniczno-Informatyczna
ul.Chodkiewicza 30, 85-064 Bydgoszcz
email: prophetnes@wp.pl

Streszczenie: *Artykuł porusza dwa zagadnienia. Pierwsze określane jest jako problem komiwojażera popularnie nazywanego TSP (z ang. Traveling Salesman Problem), oraz systemy mrówkowe (z ang. Ant Systems) jako przedstawiciel nowatorskiego podejścia do rozwiązywania problemów optymalizacyjnych z grupy NP-trudnych. Problem TSP jest zagadnieniem optymalizacyjnym polegającym na znalezieniu drogi o najmniejszym koszcie dla wyznaczonej przez komiwojażera trasy. Systemy mrówkowe są to algorytmy wzorujące się na świece przyrody, a konkretniej na sposobie organizacji kolonii mrówek w poszukiwaniu najkrótszej drogi z mrowiska do pokarmu i z powrotem. Artykuł ma za zadanie zapoznać czytelnika z dwoma zakreślonymi powyżej zagadnieniami, zaprezentować zastosowanie systemów mrówkowych do rozwiązania TSP, zbadać efektywność algorytmów mrówkowych oraz algorytmów klasycznych w poszukiwaniu optimum dla określonych problemów TSP oraz przedstawić otrzymane wyniki wraz z wnioskami końcowymi. Dodatkową częścią artykułu są kierunki dalszych badań, jakie są podejmowane przez naukowców, przy wykorzystaniu filozofii systemów mrówkowych.*

Słowa kluczowe: *Problem komiwojażera, systemy mrówkowe, algorytmy mrówkowe, algorytmy klasyczne, rozwiązanie optymalne, mrówka, jako sztuczny agent, feromon, parowanie feromonu, acotsp, concorde, tsplib, swarm-bots.*

Ant Systems applied to solve the Travelling Salesman Problem

Abstarct: *The paper discusses two problems. The first one is known as the Travelling Salesman Problem (TSP), whereas the second one is defined as the Ant Systems being the representative of innovative attitude to solving optimization problems belonging to the NP-hard group. The TSP problem is an optimizing issue that consists in finding the lowest cost travelling way for the route specified by the travelling salesman. The ant systems are algorithms pattered after the nature, more specifically, after the way an ant colony is organised is order to find the shortest way from the anthill to food and back. The aim of the paper is to familiarize readers with the above two problems, to present application of ant systems to solve the TSP, to examine efficiency of ant algorithms and classical algorithms when searching for the optimum for specific TSP problems as well as to present obtained results together with final conclusions. As an additional part of this paper, the author presented further directions of research undertaken by scientists using philosophy of ant systems.*

Keywords: *The travelling Salesman Problem, ant systems, ant algorithms, classical algorithms, optimal solution, an ant as a false agent, pheromone, pheromone evaporation, acotsp, concorde, tsplib, swarm-bots.*

1. WSTĘP

Z własnego doświadczenia wiem, że niewiele osób miało styczność z problemem komiwojażera, lub systemami mrówkowymi, a dziwnie brzmiące połączenie tych dwóch terminów, które zawierają się w tytule tego artykułu powoduje, iż przeciętny odbiorca wpada w lekkie osłupienie dziwiąc się, że nic z przekazanej mu treści nie rozumie. Na pierwszy rzut oka można faktycznie stwierdzić, że temat jest bardzo trudny, jednak rzeczywistość jest zupełnie inna. Charakterystyka obydwu zagadnień, która zarazem prowadzi do zrozumienia samej ich istoty, jest raczej prosta. Tym artykułem chciałbym zachęcić każdego do zapoznania

się z wspomnianymi powyżej zagadnieniami, które mają bardzo dużo znaczenie w dzisiejszym świecie, który przez przeciętnego Kowalskiego pojmowany jest w ujęciu globalnym, gdzie wartością jest mobilność, szybkość, skuteczność. Wszystkie te rzeczy ułatwić nam może komiwojażer wpierany przez kolonię mrówek.

2. WYJAŚNIENIE ZAGADNIENIA PROBLEMU KOMIWOJAŻERA

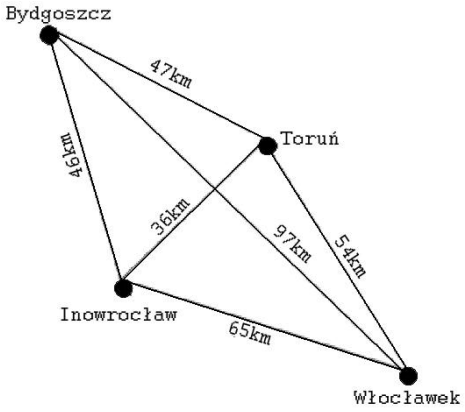
Travelling Salesman Problem w skrócie TSP, jest to źródłowa nazwa problemu znanego w Polsce, jako problem wędrującego komiwojażera. Zaliczany jest on do problemów NP – trudnych do rozwiązania z powodu dużego skomplikowania w obliczeniach, które mogą trwać nawet wiele lat. Główną ideę samego zagadnienia można sformułować następująco:
Przyjmij liczbę miast i podaj koszt podróży z dowolnego miasta do innego dowolnego miasta, a następnie określ jaki jest najmniejszy koszt, oraz najkrótsza droga tak, aby każde miasto zostało odwiedzone tylko raz i zakładając, że miastem ostatnim jest miasto z którego wyruszyliśmy.
Zależy nam na tym, aby wyznaczoną przez nas trasę pokonać przy jak najmniejszym koszcie, która określana jest terminem trasy optymalnej. Koszt takiej podróży może być rozumiany na wiele sposobów jak np.: 1) najkrótsza droga z miasta A do miasta docelowego B, 2) jak najmniejszy koszt spalania benzyny dla danej trasy, który może być uzyskany nie tylko w przypadku trasy najkrótszej, ale na przykład trasy dłuższej bez korków ulicznych. Są to tylko dwa proste przykłady, które zawierają się w samym zagadnieniu TSP.
TSP ma bardzo długą historię. Jego początki według Z. Michalewicza oraz D.B. Fogel [2] sięgają roku 1759, gdzie problem został wspomniany przez wybitnego w tamtych czasach szwajcarskiego matematyka i fizyka Leonarda Eulera. W tamtym okresie problem ten nie był rozpoznawany jako TSP, występował on pod inną nazwą. Problem dotyczył trasy konika szachowego, a dokładnie polegał on na podaniu takiej trasy konika, w której odwiedzi on wszystkie 64 pola szachownicy dokładnie raz [2].
Matematyczne problemy związane z TSP zostały również poruszone w 1800 roku przez dwóch naukowców: szkockiego matematyka Ser Williama Hamiltona oraz brytyjskiego matematyka Thomasa Kirkmana. Rozważania na temat wczesnych prac Hamiltona i Kirkmana, oraz Eulera możemy znaleźć w pracy Graph Teory 1736-1936. W tej pracy przedstawiono ciekawy problem miasteczka Königsberg (Królewiec), w którym jak to żartobliwie napisano: mieszkańcy jako rozrywkę próbują odnaleźć trasę dookoła miasta w taki sposób, aby przekroczyć każdy z siedmiu mostów tylko raz [3]. W polskiej literaturze problem ten jest znany pod nazwą problemu mostów królewieckich. Według wspomnianej pracy mieszkańcy miasteczka próbowali znaleźć rozwiązanie tego problemu, jednak jak się ostatecznie okazało bez skutku. Próby rozwiązań były dokonywane przed rokiem 1730, a wiec można stwierdzić, że pewna forma TSP była już znana przed 1759 rokiem, nie jak podaje książka Michalewicza oraz Fogel. W 1736 roku słynny matematyk tamtych czasów Euler zgłębił problem, oraz zaproponował pewien algorytm. Ostatecznie okazało się jednak, że rozwiązanie tego problemu nie jest możliwe ze względu na nieparzystą liczbę wyjść mostów zarówno na oba brzegi rzeki jak i na wyspy [4].
Początki TSP były trudne, jak każda nowa nauka. W latach 20, dwudziestego wieku, matematyk i ekonomista Karl Menger przedstawił to zagadnienie swoim kolegom na swojej uczelni w Vienna. Pierwsze konkretne nazewnictwo problemu TSP terminem „komiwojażer" nastąpiło w latach 30 dwudziestego wieku w Vienna i na Harwardzie przez niemieckiego naukowca Karl Mengera. W późnych latach 30, problem ponownie pojawił się w matematycznych kręgach uczelni Princeton. W latach 40, problem był studiowany przez statystyków i matematyka Merill Flood, którzy to spopularyzowali problem pośród swoich kolegów w korporacji RAND. Ostatecznie, problem TSP zyskał zły rozgłos, jako rodzaj problemu bardzo trudnego do rozwiązania. Sprawdzanie ścieżek jedna po drugiej nie było brane pod uwagę z powodu zbyt dużej liczby możliwych rozwiązań, innych alternatywnych propozycji rozwiązań tego problemu nieznano w tamtych czasach.
Wysokie skomplikowanie problemu przedstawia poniższa tabela, która pokazuje złożoność problemu, rozumiana, jako liczba wszystkich możliwości do rozpatrzenia w stosunku do podwojonej liczby miast, tzn. n!/(2n) = (n-1)!/2.

Tabela 1. Wielkość problemu [5]

Miast	Wielkość problemu
3	**1**
5	**12**
7	**360**
10	**181440**
15	**43589145600**
20	**60822550204416000**
25	**310224200866620000000000**
30	**4420880996869850000000000000000**

W późniejszym czasie problem był rozpowszechniany przez Hassler Whitney oraz Merrill M. Flood na uczelni Princeton. Osiągnięcia K. Mengera oraz H. Whitney'a zostały umieszczone w opracowaniu A. Schrijver On the history of combinatorial optimization, które zawiera historię problemów optymalizacyjnych, dotyczących również TSP do lat 60.

Aby zwizualizować ten problem na bazie naszych realiów poniżej można zobaczyć rysunek przedstawiający problem czterech miast województwa Kujawsko-Pomorskiego. Zadaniem jest znalezienie jak najkrótszej trasy, przy założeniu, że miastem startowym jest Inowrocław. które przedstawiają kombinację wszystkich możliwych rozwiązań danego problemu.



Rys. 9 Graf ważony przestawiający cztery polskie miasta

Całkowitą liczbę wszystkich możliwych rozwiązań powyżej przedstawionego przykładu, który reprezentuje zagadnienie TSP. można scharakteryzować, jako cykl Hamiltona, który jest reprezentowany wzorem:

$$\frac{1}{2}\cdot(n-1)!\rightarrow n>2 \qquad (1)$$

Za pomocą tego wzoru można obliczyć wielkość problemu dla dowolnej liczby miast. co czyni zagadnienie TSP bardzo trudnym do rozwiązania. Dla naszego przykładu czterech miast całkowita ilości możliwych rozwiązań jest równa ½ * 3!, czyli 3 rzeczywiste rozwiązania dla 6 dostępnych możliwych tras. Dużą zaletą jest to, iż wzór na cykl Hamiltona wyklucza powtórzenie się tego samego rozwiązania.

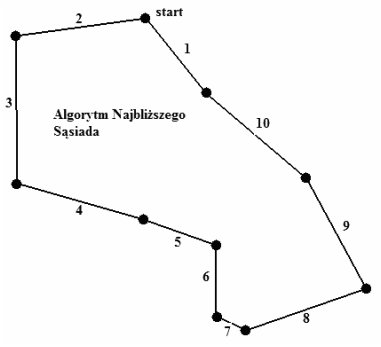
Za pomocą tego wzoru można obliczyć wielkość problemu dla dowolnej liczby miast. co czyni zagadnienie TSP bardzo trudnym do rozwiązania. Dla naszego przykładu czterech miast całkowita ilości możliwych rozwiązań jest równa ½ * 3!, czyli 3 rzeczywiste rozwiązania dla 6 dostępnych możliwych tras. Dużą zaletą jest to, iż wzór na cykl Hamiltona wyklucza powtórzenie się tego samego rozwiązania.

Tabela 2 Możliwe rozwiązania cyklu Hamiltona dla 4 przyjętych miast

Trasa	Dystans [km]
Inowr., Bydgoszcz, Toruń, Włocławek, Inowr.	**212**
Inowr., Włocławek, Toruń, Bydgoszcz, Inowr.	**212**
Inowr., Bydgoszcz, Włocławek, Toruń, Inowr.	**233**
Inowr., Toruń, Włocławek, Bydgoszcz, Inowr.	**233**
Inowr., Toruń, Bydgoszcz, Włocławek, Inowr.	**245**
Inowr., Włocławek, Bydgoszcz, Toruń, Inowr.	**245**

Tak jak można łatwo zauważyć tabela numer 2 prezentuje wszystkie możliwe rozwiązania. Cykl Hamiltona danego grafu, po którym się poruszamy, mówi nam jednoznacznie ile jest możliwych dostępnych rozwiązań pokonania założonej drogi. Jest to informacja wyjściowa do podania optymalnego rozwiązania TSP.

Tak jak wspomniano na samym początku sama charakterystyka problemu jest stosunkowo prosta, natomiast samo rozwiązanie niestety już nie. Aby sprostać problemowi TSP, wymyślono wiele różnych algorytmów, określanych dzisiaj terminem klasyczne, za pomocą których możliwe jest poszukiwanie rozwiązań optymalnych lub rozwiązań bliskich optymalnym. Rozróżniono dwie główne grupy algorytmów, do których zaliczamy: 1) algorytmy znajdujące dokładne rozwiązanie (algorytmy czasochłonne) oraz 2) algorytmy nie w pełni optymalne nazywane heurystycznymi. Pierwsza grupa są to algorytmy, które sprawdzają się tylko dla małych problemów. Do przykładowych algorytmów możemy zaliczyć m.in. *branch-and-bound, branch-and-cut*. Druga grupa to algorytmy znajdujące rozwiązania bliskie optymalnemu w stosunkowo krótkim czasie. Sprawdzają się one nawet dla dużych problemów. Do przykładowych algorytmów możemy zaliczyć m.in. algorytm najbliższego sąsiada (z ang. *Nearest Neighbour*), algorytm zachłanny (z ang. *Grendy algorithm*), algorytm drugiego przybliżenia 2-opt, k-opt, jak i również najskuteczniejszy ze znanych w dzisiejszej dobie algorytmów klasycznych Lin-Keringhana.



Rys. 10 Metoda najbliższego sąsiedztwa

Tak jak można łatwo zauważyć, TSP jest bardzo ważnym zagadnieniem, którego zastosowanie może ułatwić ludziom życie. W dzisiejszej dobie coraz popularniejsze są geolokalizatory z funkcją znajdowania najkrótszej trasy, takie jak np. map24 z portalu interia lub choćby docelu z wp.pl. Bez wątpienia można stwierdzić, iż każdy podróżujący niejednokrotnie sprawdził w jednym z podanych powyżej serwisów, trasę, jaką mógłby pojechać, aby zaoszczędzić choćby przysłowiową złotówkę. Choć ich działanie pozostawia wiele do życzenia, to ciągłe prace nad doskonaleniem zagadnienia TSP mogą nam dać świetny rezultat w postaci dobrej optymalnej trasy do miejsca naszych wymarzonych wakacji.

3. WYJAŚNIENIE ZAGADNIENIA SYSTEMÓW MRÓWKOWYCH

Pracowita mrówka, niewielkie stworzenie, które może dokonać wielkich rzeczy. Czy aby na pewno tak jest? Jak się okazuje nie do końca. Poniższy cytat w bardzo dobry sposób prezentuje to zagadnienie:

A single ant or bee isn't smart, but their colonies are. The study of swarm intelligence is providing insights that can help humans manage complex systems, from truck routing to military robots [6].

Pojedyncza mrówka pomimo swej prostoty, jako jednostka wraz z całą społecznością mrówek potrafi prezentować bardzo wysokie zorganizowanie w pracy. Jako kolonia mrówek, mogą dokonać rzeczy, które przekraczają zdolność pojedynczego osobnika. Pojęcie systemów mrówkowych można zdefiniować, jako system informatyczny, który realizuje algorytmy równolegle. Działają one na analogicznych zasadach, co odpowiadające im procesy w świecie rzeczywistym. Wiele z nich bazuje na procesach dobrze poznanych przez zoologów, czy fizyków molekularnych.

Systemy mrówkowe to pojęcie wydawałoby się zagadkowe nie mające za dużo wspólnego z informatyką i algorytmami, a jednak jest wręcz przeciwnie. Tak naprawdę tego typu podejście nie jest nam obce w dzisiejszych czasach. Wielu naukowców wzoruje się na naszej matce naturze jak na obiekcie natchnienia, wzorca dobrego do naśladowania. Nie jest to sytuacja dziwna, w końcu matka natura miała miliony lat, aby wykształcić metody rozwiązań wielu problemów. Powracając do sedna sprawy, jak sama nazwa wskazuje w tym zagadnieniu uczestniczą mrówki, dobrze zorganizowane, pracowite organizmy, potrafiące o dziwo i tu uwaga „znaleźć optymalne rozwiązanie dla określonej drogi". Drogą z punktu widzenia mrówek jest trasa prowadząca od gniazda mrówek do pożywienia. Jak to się zdarza w naturze i w jej wytworach, często na drodze takiej mrówki staje przeszkoda. Najprostszym sposobem prezentacji zagadnienia systemów mrówkowych będzie przedstawienie mikroświata w postaci gniazda, pożywienia, przeszkody oddzielającej gniazdo od pożywienia oraz dwóch dróg prowadzących do pożywienia. Celem mrówek jest dojście do pożywienia, oraz powrót do gniazda w jak najszybszym czasie, czyli wybranie trasy najkrótszej. Z obserwacji mrówek wiemy, że po pewnym czasie znajdują one krótszą z dwóch dostępnych tras. Należy sobie zadać pytanie: jak to jest możliwe? Mrówki wydzielają pewną substancję, która nazywana jest feromonem. Wędrując od gniazda do miejsca pokarmu mrówki zostawiają tzw. ślad feromonowy

Rzeczą istotną jest to, iż pozostawiony ślad feromonowy jest intensywniejszy na drodze częściej uczęszczanej. W tym momencie można zauważyć sposób, w jaki mrówki odnajdują najkrótszą trasę. Kierują się one intensywnością feromonu, im intensywniejszy tym droga częściej uczęszczana, im częściej uczęszczana oznacza to, że do pokarmu jest bliżej, a bliżej oznacza lepiej i jest rozwiązaniem optymalnym. Zobaczmy, jak to wygląda na rysunku z dwoma dostępnymi drogami. Tak jak można zauważyć mamy pokazaną drogę mrówek z gniazda do pokarmu. Do pokarmu prowadzą dwie drogi, które powstały za przyczyną głazu.

Rys. 11 Dwie drogi do pokarmu

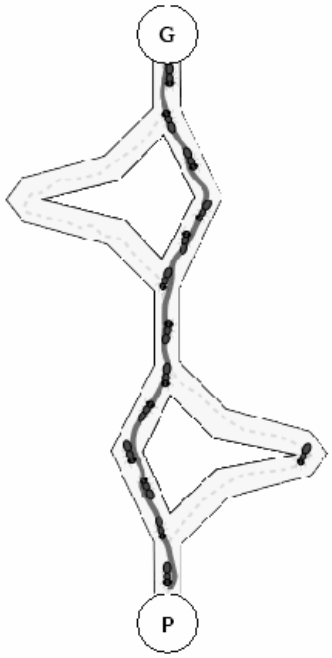
Obydwie drogi są uczęszczane w dwóch kierunkach, jako że mrówki idą po pokarm, a następnie wracają z nim do mrowiska. Na pierwszy rzut oka można zauważyć, która z dróg jest krótsza, a która dłuższa, jednak mrówki tego nie wiedzą. Tak jak już wspomniałem powyżej, im droga dłuższa, tym jest mniej uczęszczana z powodu mniejszej intensywności feromonu, im droga krótsza tym częściej uczęszczana i intensywność feromonu jest wyższa.
Idąc dalej tym tropem, należałoby sobie zadać pytanie: kiedy mrówki dojdą do tego, którą drogą należy iść, aby dotrzeć i wrócić szybciej? Wszystko to zależy od ilości mrówek, jakie wybrały określoną drogę, oraz długości trasy. Na początku mrówki losowo wybierają drogę. Oczywiste jest, że krótszą drogą mrówki szybciej wrócą z pokarmem, co spowoduje szybsze nanoszenie feromonu i wzrost jego intensywności. Jednak tak naprawdę nie da się określić, kiedy zostanie wybrana optymalna trasa. Na pewno można stwierdzić, że prędzej czy później zostanie ona wybrana. Szybkość w dużej mierze będzie zależała od szczęścia. Mam tu na myśli sytuację losową, jeśli mrówek będzie więcej i dobrze wybiorą one trasę na samym początku, znalezienie optymalnej drogi zajmie mniej czasu, natomiast gdy tych mrówek będzie mniej, znalezienie rozwiązania optymalnego w krótkim czasie będzie mało prawdopodobne.

Pojęcie algorytmów mrówkowych zostało zdefiniowane przez Marco Dorigo w 1992 roku, jako wielo-agentowy system zainspirowany przez obserwację zachowań prawdziwych kolonii mrówek wykorzystujący zagadnienie *stigmergy (STY)* (pośrednia komunikacja). Dziedzina „algorytmów mrówkowych" wywodzi się z obserwacji naturalnego środowiska życia mrówek. Pojęcie STY pełni ważną rolę w algorytmach mrówkowych, ponieważ dzięki implementacji tzw. zmiennych STY możliwe jest ich funkcjonowanie. Zmienne te zawierają informację używane przez sztuczne mrówki do komunikacji pośredniej. Dla przykładu, w algorytmie ACO zmienne STY są specjalnie zdefiniowane i używane przez mrówki do zmiany sposobu budowania rozwiązania dla rozważanego problemu.

Tabela 3 Algorytmy ACO w chronologicznej kolejności ich pojawienia się [7]

Algorytmy ACO	**Dla TSP**	**Twórcy**
System Mrówkowy AS	TAK	Dorigo 1992
Elitarny System Mrówkowy	TAK	Dorigo 1992
Ant-Q	TAK	Gambardella i Dorigo 1995-96
System Kolonii Mrówek ACS	TAK	Dorigo i Gambardella 1997
Max-Min AS	TAK	Stützle i Hoos 1996, 2000
System oceny AS_{rank}	TAK	Bullnheimer, Hartl i Strauss 1997-99
ANTS	NIE	Maniezoo 1999
Hyoer Cube AS	NIE	Blum Roli i Dorigo 2001, Blum i Dorigo 2004

Ten model został potraktowany, jako źródło inspiracji do zaprojektowania nowatorskich algorytmów. Algorytmy były tworzone z myślą zastosowania ich do problemów optymalizacji i problemów kontroli zaburzeń. Wdrożonym pojęciem wykorzystywanym przy algorytmach mrówkowych jest sztuczny agent (z ang. *artificial agent*), który jest odpowiednikiem zwykłej mrówki. Sztuczne mrówki w rozumieniu programistycznych razem współpracują w celu rozwiązywania problemów obliczeniowych. Poniżej możemy zobaczyć wykaz algorytmów mrówkowych wraz z datą ich ukazania się.



Rys. 12 Optymalna trasa mrówek

Optymalizacja kolonią mrówek (z ang. *Ant Colony Optimization,* ACO) – jest jednym z najbardziej znanych w obecnych czasach algorytmów mrówkowych. Został

on po raz pierwszy zdefiniowany przez Marco Dorego, Di Caro oraz Gambardella w 1999 roku jako technika dla rozwiązywania problemów dyskretnej optymalizacji. ACO został przedstawiony, jako algorytm, który może znaleźć dobrą ścieżkę poprzez graf. Jest to algorytm inspirowany teorią *foragingu* [8] dla kolonii mrówek jak i problemów dyskretnej optymalizacji. Algorytm jest przeznaczony do rozwiązywania dwóch rodzajów problemów: a) optymalizacyjnych statycznych, czyli problemów gdzie podczas pracy żadne dane się nie zmieniają i tu przykładem może być TSP, jak i b) problemów dynamicznych, gdzie informacje podczas działania programu zmieniają się i tu przykładem może być problem wyznaczania trasy, w którym parametrem dynamicznym jest ciągle zmieniający się ruch. Podstawą do rozpoczęcia badań nad algorytmami mrówkowymi było samo obserwowanie środowiska mrówek. Zaobserwowaną ciekawostką było to, iż w przeważający sposób mrówki porozumiewają się ze sobą za pomocą substancji chemicznych przez nie produkowanych. Jak już wspomniałem bardzo ważną rzeczą w tym algorytmie jest pośrednia komunikacja *foraging*, reprezentowana przez ślad feromonowy. Parowanie tego feromonu ma tą przewagę, że może zapobiegać konwergencji (zbieżności) dla optymalnych lokalnych rozwiązań. Gdybyśmy założyli, że nie istnieje zagadnienie parowania, to za każdym razem każda ze ścieżek wybierana przez pierwszych sztucznych agentów była by traktowana jednakowo, byłaby jednakowo atrakcyjna, co nie miałoby zastosowania do rozwiązywania problemów optymalizacyjnych. W ten sposób, kiedy jedna mrówka znajdzie dobrą ścieżkę z kolonii do źródła pokarmu, pozostałe mrówki przychylają się bardziej do tej trasy. Pomysłem dla algorytmu ACO jest naśladowanie tego zachowania za pomocą sztucznych agentów poruszających się w obrębie grafu w celu rozwiązania danego problemu. Algorytm ACO został użyty do rozwiązania problemu TSP. Algorytm ten ma przewagę nad algorytmami genetycznymi lub algorytmem symulowanego wyżarzania. Ważną jego cechą jest to, że dla grafu zmieniającego się dynamicznie, algorytm ACO może pracować ciągle i zaadaptować się do zmian w czasie rzeczywistym. Zasługą takiego działania jest metoda wyznaczanie trasy w sieci (z ang. *network routing)* i systemy miejskiego transportu (z ang. *urban transportation systems)*. Poniżej możemy zobaczyć pseudo kod algorytmu ACO:

Kod 1 Pseudo Kod algorytmu ACO [9]

```
1  procedure ACO_MetaHeuristic
2     while(not_termination
3        generateSolutions()
4        pheromoneUpdate()
5        daemonActions()
6     end while
7  end procedure
```

Gdzie poszczególne funkcje mają poniższe zadania.

generateSolutions() – zarządza kolonią mrówek, które jednocześnie i asynchronicznie odwiedzają sąsiadujące stany danego rozważanego problemu przez przesuwanie się poprzez sąsiednie wierzchołki, opisanego problemu, danego grafu G. Przesuwanie mrówek odbywa się przy pomocy stochastycznie lokalnie podejmowanych decyzji, które są podejmowane na bazie śladów feromonowych i innych informacji heurystycznych. W ten sposób mrówki budują w sposób przyrostowy rozwiązanie i oceniają cząstkowe rozwiązania, które będą wykorzystane przez funkcję pheromoneUpdate(), aby zdecydować ile feromonu zostanie pozostawione,

pheromoneUpdate() – jest to funkcja, za pomocą której modyfikowany jest ślad feomonowy. Ten ślad może wzrastać, dzięki pracy mrówek, które pokonują kolejne krawędzie (wierzchołki), jak i może się zmniejszać, przez naturalne zaimplementowane parowanie. Parowanie feromonu zapobiega skupieniu się na jednym obszarze przeszukiwań, a zarazem zachęca do przeszukiwania nowych do tej pory nieodwiedzanych miejsc,

daemonActions() – jest to funkcja, która jest używana do implementacji zcentralizowanych działań, które nie mogą być przeprowadzone przez pojedynczą mrówkę. Przykładem działania tej funkcji jest włączenie procedury lokalnej optymalizacji, lub zbieranie globalnych informacji, które mogą być wykorzystane do podejmowania decyzji czy użytecznym jest, lub też nie pozostawienie dodatkowego feromonu, aby wpłynąć na proces przeszukiwania z poza lokalnej perspektywy.

Do dwóch najważniejszych wzorów, które opisują zagadnienie systemów mrówkowych należą:

Wybór ścieżki – mrówka przebędzie drogę z punktu *i* do punktu *j* z prawdopodobieństwem równym:

$$p_{ij} = \frac{\tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{\sum \tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}} \qquad (2)$$

gdzie

$\tau_{i,j}$ - ilość feromonu na ścieżce *i,j,*

α - parametr do kontroli wpływu $\tau_{i,j}$,

$\eta_{i,j}$ - określa atrakcyjność ścieżki *i,j,*

β - parametr do kontroli wpływu $\eta_{i,j}$.

Uaktualnienie feromownu – to zagadnienie reprezentowane jest wzorem

$$\tau^{(n)}{}_{i,j} = p\tau^{(s)}{}_{i,j} + \Delta\tau_{i,j} \quad (3)$$

gdzie wskaźnik n oznacza normę, zaś s – stan, ilość fermonu

$\tau_{i,j}$ - ilość feromonu na ścieżce *i,j,*

p – skala parowania feromonu,

$\Delta\tau_{i,j}$ - oznacza ilość feromonu pozostawionego, określane wzorem

Poniżej można zobaczyć bardziej szczegółowy pseudo kod jednego z wielu algorytmów mrówkowych, określanego jako ACS (z ang. *Ant Colony System*), czyli optymalizacja kolonia mrówek.

Kod 2 Pseudo kod algorytmu kolonii mrówek ACS [10]

```
1  Initialize
2  Repeat {
3   Place each ant in a randomly chosen city;
4   For each ant
5     Repeat {
6       Choose NextCity (each ant);
7       Update pheromone levels using a local rule;
8     } Until (No more cities to visit);
9     Return to the initial cities;
10    Compute the length of the Tour found by each ant;
11  End For;
12  Update pheromone level using a global rule;
13 }
14 Print Best Path;
```

Podsumowując można zatem wyróżnik kilka etapów cyklu życia mrówki w systemie poszukującym optymalnego rozwiązania:

- Wybór losowego miasta.
- Miasto zostało wybrane.
- Konstruowanie rozwiązania. Wybór kolejnego miasta.
- Powrót do miasta startowego.
- Po zakończeniu przez wszystkie mrówki swoich tras następuje pozostawienie feromonu na ścieżce, którą przebyły.

4. ZASTOSOWANIE SYSTEMÓW MRÓWKOWYCH DO ROZWIĄZANIA PROBLEMU KOMIWOJAŻERA

Drogi czytelniku, posiadasz już podstawową wiedzę z zakresu problemu TSP oraz systemów mrówkowych (SM). Najwyższy już czas, aby połączyć te dwa zagadnienia i przedstawić rezultaty zastosowania koloni mrówek w odnajdowaniu optymalnej trasy. Celem tej części artykułu jest przedstawienie zależności pomiędzy tymi dwoma zagadnieniami, aby zrozumieć przyczynę zastosowania SM do TSP, oraz innych łączących je elementów.

TSP jest problemem bardzo starym. Doczekał on się wielu badań, znaleziono wiele rozwiązań, mniej lub bardziej efektywnych, poświęcono mu bardzo dużo miejsca w literaturze. Z punktu widzenia zdobytej wiedzy na ten temat można powiedzieć, że mrówki zyskały dużo korzyści dzięki TSP, a TSP zyskało równie dużo korzyści dzięki mrówką. TSP pełniło bardzo dużą role w algorytmie optymalizacji kolonią mrówek (ACO). TSP było bardzo ważne, ponieważ jako problem NP-trudny, posłużył on do pierwszych testów funkcjonowania systemów mrówkowych (AS). Sami autorzy wymieniają kilka przyczyn, dla których zdecydowali się wybrać ten, a nie inny problem do przedstawienia funkcjonowania algorytmu ACO. Do takich powodów zostały zaliczone m.in.:

TSP jako problem NP-trudny ma szerokie zastosowanie,
W dosyć łatwy sposób można zaimplementować algorytmy ACO do tego problemu,
Kod jest łatwy do zrozumienia, dzięki TSP samo serce ACO nie jest zasłaniane dodatkowymi technicznymi sprawami, które mogłyby skomplikować sam kod i zmniejszyć przejrzystość ACO.

Tak jak powyżej wspomniałem pierwszy system mrówkowy był sprawdzany na problemie TSP. Początkowo wyniki testów AS na bazie TSP nie były zbyt obiecujące, jednak po upływie pewnego czasu sama filozofia wykorzystania mrówek zaczęła się rozwijać i powstawały kolejne rozszerzenia AS, mam tu na myśli m.in. Elitist Ant Systems, MMAS, czy AS_{rank}. Wszystkie te rozwinięcia pierwszego bazowego AS, spowodowały bardzo duże usprawnienie systemu i wzrost jego wydajności. Na chwilę obecną algorytmy ACO uznawane są za jedne z najbardziej skutecznych, efektywnych algorytmów zajmujących się problemami optymalizacyjnymi, które rozwiązują problem TSP, oraz wiele innych trudnych problemów. Jeśli chodzi o różnice pomiędzy bazowym AS a rozszerzeniami to dotyczą one

sposobu uaktualniania śladu feromonowego, jak i również pewne dodatkowe rzeczy związane z zarządzaniem tymi śladami. Oprócz algorytmów, rozszerzeń AS, które nie wniosły dużych zmian, w późniejszym okresie zaproponowano również algorytmy, które wprowadziły większe zmiany. Do tych algorytmów możemy zaliczyć systemy kolonii mrówek (ACS), Ant-Q, ANTS *(z* ang. *Approximate Nondeterministic Tree Search)* wykorzystujące techniki dolnej granicy w programowaniu, czy środowisko hyper-cube dla ACO.

Jeszcze jedną rzeczą, o której należy wspomnieć, jest to zastosowanie algorytmów lokalnego przeszukiwania, które w znaczący sposób przyśpieszyły jego działanie i pozwoliły na osiągnięcie lepszych rezultatów w rozwiązaniach TSP. Do stosowanych algorytmów zaliczane są: 2-opt, 2,5-opt jak i najskuteczniejszy 3-opt. W książce *Ant Colony Optimization* Dorigo i Stützle można znaleźć wiele badań dotyczących tego zagadnienia, pokazujących skuteczność wszystkich trzech sposobów przeszukiwania w oparciu o dwa wybrane problemy z bazy TSPLIB.

Tak jak można zauważyć TSP był pierwszym problemem kombinatorycznej optymalizacji, który posłużył do sprawdzania skuteczności funkcjonowania algorytmów ACO. Pierwszym testowanym algorytmem był AS, który uzyskał dobre rezultaty, ale tylko dla problemów TSP o małej wielkości. Przy problemach o większej ilości miast wyniki były dosyć słabe. Głównym celem zastosowania AS dla TSP było potwierdzenie przez Dorigo koncepcji zastosowania tego typu algorytmów do rozwiązywania problemów optymalizacyjnych. Na chwilę obecną dostępna jest dosyć duża gama algorytmów ACO. Można powiedzieć, iż te algorytmy wniosły duży wkład w rozwój rozwiązywania trudnych problemów i osiągnęły bardzo dobre rezultaty niejednokrotnie znajdując najlepsze rozwiązania danego problemu.

5. BADANIE EFEKTYWNOŚCI ALGORYTMÓW

W tej części zostanie przedstawiony sposób przeprowadzenia części eksperymentalnej, której celem jest dokonanie porównania efektywności algorytmów mrówkowych w stosunku do algorytmów klasycznych. Zostanie dokonane porównanie efektywności oprogramowania ACOTSP oraz CONCORE, jako dwóch programów mogących poradzić sobie z problemem NP – trudnym, jakim jest TSP. Będzie to również konfrontacja algorytmów klasycznych z całkowicie nowym podejściem, jakim jest naśladowanie koloni mrówek do rozwiązania problemów optymalizacyjnych.

Testy badania skuteczności rozwiązań zostały przeprowadzone na komputerze klasy PC o ogólnej specyfikacji:

- Procesor: AMD Athlon Dwurdzeniowy 2 x 2,6 GHz,
- Pamięć RAM: 2 GB,
- System operacyjny: Windows Vista Premium.

5.1. Charakterystyka danych wejściowych

Program CONCORDE nie potrzebuje żadnych dodatkowych parametrów, aby znaleźć rozwiązanie problemu. Jeśli chodzi o oprogramowanie ACOTSP, parametry te muszą być odpowiednio dobrane, aby rezultaty były możliwie najlepsze. Zgodnie z plikiem readme i według zaleceń autora dokumentu każdy z tych systemów uzyskuje dobre rezultaty dla innych parametrów. Nie zawsze ustawienia domyślne dają dobre rezultaty. Do najważniejszych parametrów, które mają wpływ na końcowy rezultat programu zaliczamy:

Tabela 4 Wyróżnienie głównych parametrów mających wpływ na końcowe rozwiązanie problemu

L.p.	Opis parametru	Komenda
1	Liczba uczestniczących mrówek	-m / --ants
2	Parametr określający wpływ śladu feromonowego	-a / --alpha
3	Parametr określający wpływ heurystycznych informacji	-b / --beta
4	Parametr określający parowanie feromonu	-e / -rho
5	Prawdopodobieństwo najlepszego wyboru w konstruowaniu trasy	-q / --q0
6	Wybór lokalnego przeszukiwania 0: brak, 1: 2-opt, 2: 2.5-opt, 3: 3-opt	-l / --localsearch

Dla wymienionych powyżej parametrów starano się znaleźć jak najlepszą kombinację tak, aby otrzymane rozwiązanie był jak najlepsze. Testy zostały przeprowadzone dla problemu wielkości 783 miast o oznaczeniu rat783, dla którego optimum jest równe 8806. Przeprowadzono wiele testów dla każdego rodzaju algorytmu mrówkowego w celu ustalenia optymalnych parametrów, dzięki którym wynik będzie jak najbliższy rozwiązaniu najlepszemu.

5.2. Sposób przeprowadzenia eksperymentu

Dzięki ustalonym parametrom możemy przejść do porównania algorytmów klasycznych z algorytmami mrówkowymi. Ostatnią rzeczą, jaką nam pozostaje ustalić jest wybór problemów z biblioteki TSPLIB oraz ustalenie ilości tych problemów. Ustalono, iż 10 wybranych problemów pozwoli w znaczący sposób zbadać efektywności obydwóch typów algorytmów. Będą one badane według poniższych zasad:

Jeśli chodzi o ACOTSP, to dla każdego problemu program będzie uruchamiany trzykrotnie dla r = 10, natomiast CONCORDE trzykrotnie bez dodatkowych parametrów,
Skuteczność danego algorytmu będzie oceniana następująco:
przez podanie wyniku otrzymanej długości trasy,
procentowo w postaci: optimum spełnione w x%, tak jak to zostało przedstawione w tabeli numer 5,
Po zebraniu wszystkich wyników, zostanie sporządzony ogólny wykres efektywności wszystkich algorytmów. Dla każdego algorytmu zostanie zsumowana wartość „optimum spełnione w x %", przez co otrzymamy całkowitą ilość punktów na 1000 możliwych. Ten sposób prezentacji danych pomoże w łatwy sposób ocenić, jaka jest hierarchia wszystkich testowanych algorytmów na przekroju 10 zbadanych próbek.

Poniżej w tabeli zostało przedstawione 10 problemów, które zostaną poddane testom wraz z oczekiwaną wartością optimum, która będzie docelowa dla testowanych algorytmów.

Tabela 7 Lista rozpatrywanych problemów wraz z podanym optimum

Lp	Kod problemu	Opt.	Opis
1	**Eil51** (Christofides, Eilon)	426	Problem dla 51 miast
2	**D198** (Reinelt)	15780	Reprezentuje problem wywierconych dziur w materiale (z ang. *Dribling Problem*). Wielkość problemu 198 dziur
3	**Gil262** (Millet, Johnson)	2378	Problem dla 262 miast
4	**Lin318** (Lin, Kernighan)	42029	Problem dla 318 miast
5	**Pcb442** (Groetschel, Juenger, Reinelt)	50778	Reprezentuje problem wywierconych dziur w materiale (z ang. *Dribling Problem*). Wielkość problemu 442 dziury
6	**Rat783** (Pulleyblank)	8806	Problem połączonej sieci elektrycznej dla 783 punktów
7	**Pcb1173** (Juenger, Reinelt)	56892	Reprezentuje problem wywierconych dziur w materiale (z ang. *Dribling Problem*). Wielkość problemu 1173 dziur
8	**D1291** (Reinelt)	50801	Reprezentuje problem wywierconych dziur w materiale (z ang. *Dribling Problem*). Wielkość problemu 1291 dziur
9	**Nrw1379** (Bachem, Wottawa)	56638	Problem 1379 miejscowości w Nadreni Północna-Westfalia
10	**Pr2392** (Padberg, Rinaldi)	378032	Problem 2392 miast

5.3. Wnioski i podsumowanie

Po przeprowadzeniu szeregu testów zgodnie z ustalonymi zasadami z całą pewnością można stwierdzić wyższość algorytmów mrówkowych (ALM) nad algorytmami klasycznymi (ALK). Z dziesięciu badanych próbek tylko w jednym przypadku doszło do sytuacji, w której ALK, czyli algorytm Lin-Kernighan (LK) uzyskał lepszy rezultat od wszystkich pozostałych ALM. Mogło to wynikać z charakterystyki problemu, jakim jest nrw1379. W pozostałych przypadkach algorytm LK wyprzedził w najlepszym przypadku i tak stare już ALM, czyli AS oraz EAS, których początek sięga 1992 roku. Pozostałe algorytmy z grupy ALK zdecydowanie są poza czołówką, do której zaliczamy wszystkie ALM oraz algorytm LK. Spełniały one optimum w zakresie od 79% do 89%, co jest wynikiem dalekim od najlepszych.

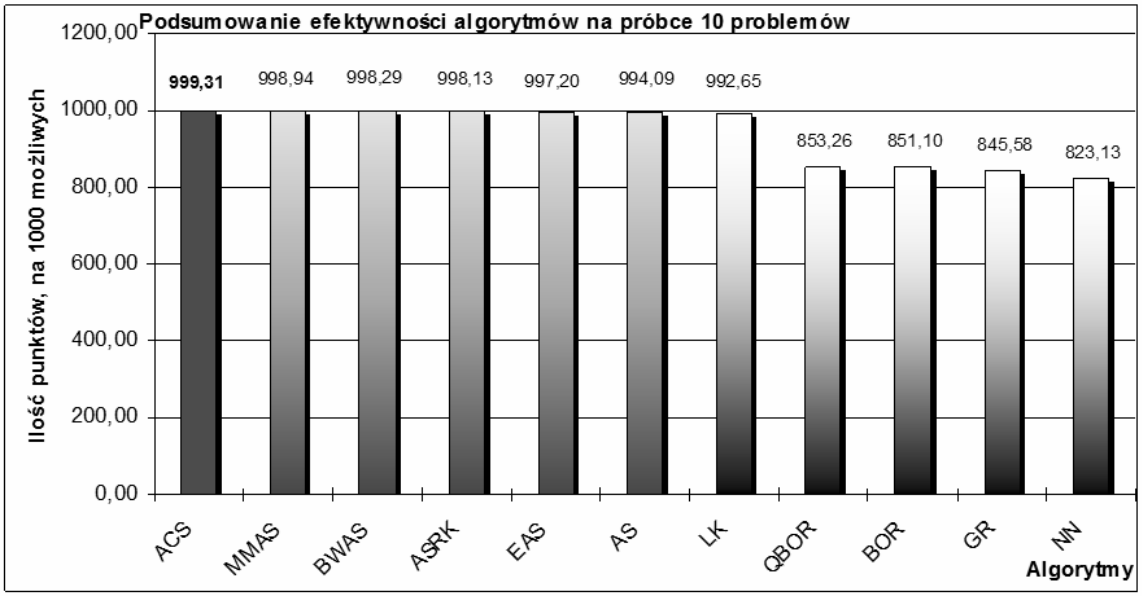
Zauważalną rzeczą jest skłonność wszystkich testowanych algorytmów do uzyskiwania gorszych rozwiązań wraz ze wzrostem wielkości problemu. Wyraźnym wskaźnikiem do tych skłonności był siódmy testowany problem pcb1173. Potwierdzeniem tego stwierdzenia był problem d1291 oraz każdy następny. Powyżej na wykresie podsumowującym wszystkie przeprowadzone testy możemy zobaczyć hierarchie wszystkich algorytmów oraz ilość otrzymany punktów. Maksymalną ilością punktów jest 1000. Najbliżej tej wartości znajduje się przedstawiciel grupy ALM, czyli ACS z wynikiem 999,31 punktu. Co warte podkreślenia pierwsze sześć miejsc pod względem efektywności poszukiwania optimum zajmują ALM. Kolejne lokaty to już przedstawiciele ALK i ich zdecydowany lider, czyli algorytm LK, który uznawany jest za najlepszy ALK do rozwiązań problemu TSP.

Marco Dorigo, który jest pomysłodawcą zastosowania zachowania mrówek do rozwiązywania problemów optymalizacyjnych, osiągnął ogromny sukces, którego potwierdzeniem jest powyższy artykuł. Algorytmy mrówkowe to nowa generacja algorytmów stosująca nowe podejście do bardzo trudnych problemów, podejście, które dało znakomite rezultaty. Algorytmy mrówkowe znajdują szereg innych zastosować nie tylko do TSP, zastosowań które są wykorzystywane w sposób praktyczny w wielu dziedzinach życia. Celem tego artykułu było zastosowanie algorytmów mrówkowych do rozwiązań problemu TSP, który od samego początku istnienia algorytmów mrówkowych miał z nimi bardzo dużo do czynienia, ponieważ był to pierwszy problem, do jakiego zastosowano owe algorytmy, aby sprawdzić ich efektywność.

wysoko zorganizowanych. Dalszym kierunkiem jaki obrali sobie naukowcy jest wykorzystanie zdobytej wiedzy na temat koloni mrówek oraz przełożenie jej do zastosowań w świecie rzeczywistym. Określeniem, które opisuje nowoczesną mrówkę, która stosuję filozofię koloni mrówek w świecie rzeczywistym jest *s-bot*. Czym jest s-bot? Są to małe roboty skonstruowane w celu wykonywania wielu skomplikowanych zadań w świecie rzeczywistym wykorzystując pracę w grupie, lub inaczej mówiąc kolonii. Podstawą to stworzenia wspomnianych robotów jest *Swarm Theory,* która brzmi:

A single ant or bee isn't smart, but their colonies are. The study of swarm intelligence [11] is providing insights that can help humans manage complex systems, from truck routing to military robots [12].

Oprogramowanie		ACOTSP						Concorde TSP				
Plik *.tsp	optimum	ACS	MMAS	BWAS	Asrank	EAS	AS	Lin Kernighan	Quick Boruvka	Boruvka	Greedy	Nearest Neighbor
		ACS	MMAS	BWAS	ASRK	EAS	AS	LK	QBOR	BOR	GR	NN
Ilość pkt na 1000		999,31	998,94	998,29	998,13	997,20	994,09	992,65	853,26	851,10	845,58	823,13
średnia %		99,93%	99,89%	99,83%	99,81%	99,72%	99,41%	99,26%	85,33%	85,11%	84,56%	82,31%



Rys. 13 Podsumowanie efektywności algorytmów

6. KIERUNKÓW DALSZYCH BADAŃ

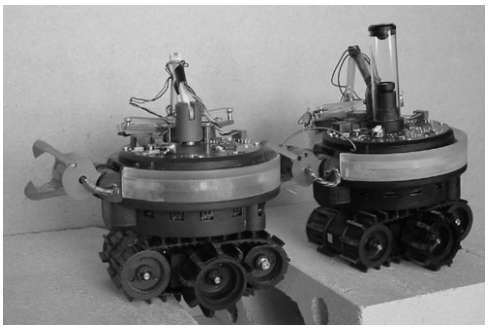
W tym punkcje zostaną przedstawione kierunki dalszych badań jakie zostały podjęte przez naukowców, wzorcujących się na koloniach mrówek jako stworzeń

Marco Dorigo był osobą koordynującą zakończony już projekt *Swarm-Bots: Swarms of self-assembling artefacts,* który był prowadzony w latach 2001-2005. Jego celem było studiowanie nowych podejść w stosunku do projektowania i wdrażania samo organizujących oraz samo-kontrolujących się robotów wynalezionych przez człowieka. Na samym początku

postawiono sobie szczegółowe cele do których zaliczamy:

- zaprojektowanie i utworzenie sprzętu w postaci: *s-bot*, *s-toy* oraz areny pna której będzie można przeprowadzać testy,
- zaprojektowani i utworzenie symulatora,
- zaprojektowani i utworzenie systemu kontroli swarm-bot'ów,
- integracja, testowanie oraz ocena rezultatu wszystkich powyższych czynności.

Założeniami projektu było utworzenie od 30 do 35 identycznych s-botów które tworzyły by kolonię. Każdy z nich jest niezależna jednostką zdolną do wykonywania podstawowych zadań takich jak: nawigacja, umiejętność percepcji otaczającego go świata oraz możliwość zaczepienia się do innego obiektu.



Rys. 14 Kooperacja grupy robotów [7]

Pojedynczy s-bot jest w stanie komunikować się z pozostałymi jednostkami oraz ma możliwość połączenia się z nimi w sposób sztywny lub elastyczny. Połączenie kilku s-bot'ów daje nam kolonię bot'ów (z ang. *swarm-bot*). Kolonia s-bot'ów jest zdolna do wykonywania badań, nawigacji i transportu ciężkich przedmiotów na bardzo szorstkim terenie, szczególnie jeśli jeden s-bot ma poważne problemy w realizacji tego zadania w pojedynkę. S-boty komunikują się ze sobą za pomocą obręczy wyposażonej w diody świecące w różnych kolorach z możliwością mrugania o różnych częstotliwościach.
Kolejnym elementem, który był wykorzystywany w powyższym projekcje jest s-toy. Jest to specjalnie skonstruowany obiekt, który może być traktowany jako zdobycz lub oznaczenie gniazda dla s-bot'ów. S-toy posiada wieżyczkę, która może być zdemontowana dzięki czemu jest możliwość obciążenia s-toy'a, maksymalnie do 2kg w celu zaimplementowania problemu współpracy w transporcie. Jest on wyposażony w identyczną obręcz jak s-bot'y co umożliwia zaczepienie s-bot'a do s-toy'a. Wieżyczka ma możliwość świecenia w dwóch kolorach. S-toy może dodatkowo emitować dźwięk w celach lokalizacyjnych.

Zostało podjętych wiele eksperymentów przy użyciu s-bot'ów, które potrafiły wykazać się dużą samoorganizacją doprowadzając dane zadanie do realizacji. Do przykładowych eksperymentów można zaliczyć m.in.:

Scenario – jest eksperymentem polegającym na udaniu się przez s-bot'y do obiektu, którym w tym przypadku jest s-toy oraz przetransportowanie go do bazy. W tym miejscu można zobaczyć analogie do systemów mrówkowych. S-toy'em może być pokarm natomiast bazą, do której ma on zostać dostarczony jest gniazdo. Jest to typowy przykład organizacji w transporcie, gdzie pojedynczy osobnik nie daje rady w realizacji zadania i gdzie grupa s-bot'ów musi się zorganizować w celu ukończenia zadania,
Przejście przez dziurę – jest to przykład sytuacji, w której pojedynczy s-bot nie da sobie rady. Kolonia s-bot'ów potrafi rozwiązać ten problem tak, aby kontynuować daną wędrówkę,
Współdziałanie w przemieszczaniu się – jest to przykład zadania, w którym s-bot'y przemieszczają s-toy'a i muszą zachować koordynację ruchów tak, aby sobie nawzajem nie przeszkadzać,

Patrząc z optymizmem na dotychczasowe dokonania naukowców i dynamizm, z jakim dziedzina swarm-bot'ów się rozwija można wybiec trochę w przyszłość pokazują ogromne zastosowanie tych urządzeń, które mogą być przeznaczone do wykonywania wielu skomplikowanych zadań na ziemi, jaki i w innych miejscach naszego układu słonecznego. Organizacja NASA, wyraziła duże zainteresowanie tego typu rozwiązaniem. Gdybyśmy wzięli pod uwagę sytuację, w której jeden z robotów podczas misji na Marsie nie mógłby kontynuować swojego zadania z różnych powodów, pozostałe s-bot'y mogłyby się same zorganizować w celu jego dokończenia. BBC news opublikowała na swojej stronie ciekawy artykuł pt: „*Smart future for swarm robots*"[13], w którym to umieszczony krótki filmik prezentujący możliwości organizacyjne s-bot'ów. Nadaną im dwa zadania, gdzie jedno miało być wykonane przez 80% dostępnych s-bot'ów a drugie przez pozostałe 20%. Przeprowadzono symulację w postaci zabierania kolejnych s-bot'ów z areny, po której roboty same się organizowały doprowadzając do ustalonej na początku proporcji 80/20. Ciekawą perspektywą zastosować kolonii s-bot'ów mogłoby być użycie ich do szukania ocalałych ludzi po jakiejś katastrofie oraz poszukiwanie innych zagrożeń, jakie mogłyby wystąpić.

Literatura

1. MICHALEWICZ, Z., FOGEL, D.B., Jak to rozwiązać, czyli nowoczesna heurystyka, Warszawa, Wydawnictwo Naukowo-Techniczne Warszawa, 2006, s. 226,
2. BIGGS, N. L., LLOYD, E. K., WILSON, R. J., Graph Theory 1736-1936, [online], Oxford, Oxford University Press, 1998
3. HELSGANN, K., NGASSA, J-L., KIERKEGAARD, J., ACO and TSP, Roskilde University, may 2007
4. MILLER, P., Swarm Theory, [online], National Geographic Staff, lipiec 2007, http://ngm.nationalgeographic.com/2007/07/swarms/miller-text.
5. DORIGO, M., STÜTZLE, T., Ant Colony Optimization, London, The MIT Press Cambridge Massachusetts Institute of Technology, 2004, p. 69
6. DORIGO, M., GAMBARDELLA, M.l., Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1997, p. 53-66,
7. PALMER, J., Smart future for swarm robots, Technology Reporter, BBC News, August 2008