MAŁOPOLSKI Waldemar[1]

Implementacja algorytmu A* do wyznaczania tras przejazdu w programie symulacyjnym Arena

Podsystemy transportowe,
Modelowanie
Symulacja,

Streszczenie

W artykule przedstawiono metodę implementacji algorytmu A za pomocą bloków funkcjonalnych w programie symulacyjnym Arena do wyznaczania trasy przejazdu pojazdów w podsystemie transportowym. W celu poprawienia efektywności działania algorytmu, zaproponowano metodę kodowania informacji opisującej dopuszczalne kierunki przejazdu pojazdów. Aby zweryfikować zaproponowane rozwiązanie zbudowano model symulacyjny i przeprowadzono badania, które potwierdziły jego efektywność.*

IMPLEMENTATION OF A* ALGORITHM TO OUTLINE ROUTES IN THE ARENA SIMULATION PROGRAM

Abstract

This paper presents a method for the A algorithm implementation using the functional blocks in the Arena simulation program to determine the routes of vehicles in the transport subsystem. In order to improve the efficiency of the algorithm, a method of encoding information describing the allowable directions of the move of vehicles was proposed. The simulation model was built to verify the proposed solution. The simulations confirmed the effectiveness of this solution.*

1. WSTĘP

Postępująca globalizacja gospodarki światowej przyczynia się m.in. do wzrostu konkurencji w sferze produkcyjnej. Efektem tych zmian jest dążenie do optymalizacji kosztów produkcji. Ma to szczególne znaczenie w krajach o wysokim koszcie pracy ludzkiej. Poszukiwane są nowe metody w zakresie zarządzania i sterowania produkcją. Jednym z kierunków badań w tym zakresie jest zastosowanie koncepcji sterowania rozproszonego, pozwalającego na budowę samokonfigurujących się zautomatyzowanych systemów produkcyjnych, posiadających wysoką elastyczność i odporność na zakłócenia [9]. W systemach produkcyjnych szczególną rolę odgrywają podsystemy transportowe zbudowane z automatycznie sterowanych pojazdów [8]. Bardzo ważna jest optymalizacja ich działania ze względu na fakt nie generowania przez czynności transportowe wartości dodanej do wyrobu finalnego.

Szczególną rolę w tych badaniach odgrywają narzędzia do modelowania i symulacji. Dzięki ich wykorzystaniu możliwe jest testowanie nowych koncepcji w zakresie metod sterowania i weryfikacja poprawności ich działania. Jednym z programów wykorzystywanych do modelowania i symulacji podsystemów transportowych jest Arena [3]. Program ten posiada specjalnie dedykowane do tego celu narzędzia. Jednak wykorzystanie tego programu do testowania specyficznych rozwiązań w zakresie podsystemów transportowych wymaga rozbudowy jego funkcjonalności.

2. MODELOWANIE PODSYSTEMÓW TRANSPORTOWYCH W PROGRAMIE ARENA

Arena jest graficznym środowiskiem do budowania za pomocą bloków funkcjonalnych modeli symulacyjnych, które są następnie tłumaczone na język SIMAN. Wyposażenie programu w narzędzia wspomagające przygotowanie danych wejściowych, następnie przeprowadzenie eksperymentu symulacyjnego i analizę wyników symulacji w znaczący sposób ułatwia pracę. Do modelowania podsystemów transportowych jest dostępny specjalny zestaw narzędzi. Dzięki niemu można budować modele podsystemów transportowych zawierających np. taśmociągi jak i dyskretne środki transportu.

Modelowanie dyskretnych podsystemów transportowych jest możliwe do zrealizowania na dwa sposoby. Pierwszy uproszczony sposób polega na definiowaniu miejsc, pomiędzy którymi mają być realizowane zadania transportowe oraz długości dróg transportowych, które ma pokonać obiekt. Prędkość ruchu jest przypisana do środka transportu, np. automatycznie sterowanego pojazdu. Czas realizacji zadania transportowego jest wyliczany na podstawie prędkości i drogi. Takie rozwiązanie jest zupełnie wystarczające w wielu prostych przypadkach.

Drugi sposób modelowania pozwala na dokładne opisanie parametrów systemu transportowego poprzez definiowanie miejsc, pomiędzy którymi mają być realizowane zadania transportowe oraz sieci transportowej, składającej się węzłów

[1]Politechnika Krakowska, Zakład Zautomatyzowanych Systemów Produkcyjnych, Al. Jana Pawła II 37, 31-864 Kraków, e-mail: małopolski@m6.mech.pk.edu.pl

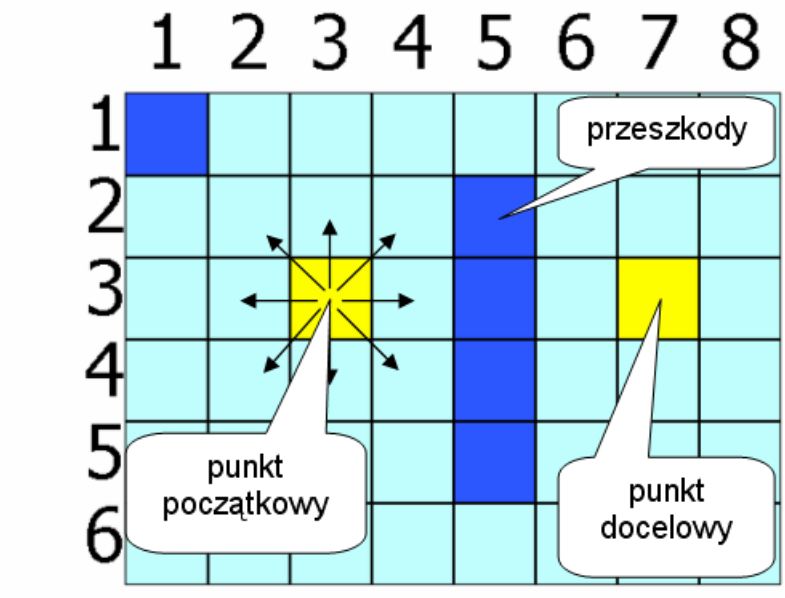
(skrzyżowań) i łączących je linii (dróg). W ten sposób jest dokładnie definiowana przestrzeń transportowa, po której poruszają się np. automatycznie sterowane pojazdy. Przed realizacją zadań transportowych program Arena wylicza tzw. macierz połączeń pomiędzy wszystkimi miejscami, które mogą być punktami początkowymi lub końcowymi poszczególnych zadań transportowych. Podczas realizacji zadania transportowego program automatycznie wybiera najkrótszą drogę przejazdu. Aby umożliwić testowanie różnych metod sterowania podsystemami transportowymi konieczna jest modyfikacja sposobu wyznaczania trasy przejazdu oraz pełna kontrola nad każdym etapem realizacji zadnia transportowego. W tym celu zaimplementowano do programu Arena algorytm A*.

3. WYZNACZANIE TRASY PRZEJAZDU W OPARCIU O ALGORYTM A*

3.1 Implementacja algorytmu za pomocą bloków funkcjonalnych

Algorytm A* jest powszechnie znaną metodą wyznaczania tras przejazdu pomiędzy dwoma dowolnie położonymi punktami [1]. Może być stosowany również w środowisku zawierającym przeszkody, które trzeba ominąć. Znane są rózne odmiany tego algorytmu [2, 4, 5, 6, 7]. Podstawowym wymogiem, warunkującym zastosowanie tego algorytmu, jest dokonanie dyskretyzacji powierzchni przeznaczonej na drogi transportowe. Dyskretyzacja może być zrealizowana za pomocą różnych kształtów geometrycznych, jednak najczęściej stosowane są kwadraty lub prostokąty o jednakowych wymiarach. Dokonanie dyskretyzacji za pomocą takich samych kwadratów (lub prostokątów) pozwala na łatwy zapis informacji charakteryzujących środowisko transportowe w postaci tablicy (macierzy). Biorąc pod uwagę względy techniczne istotne jest dobranie odpowiedniej wielkości kwadratów tak, aby gwarantowały one osiągnięcie wymaganej precyzji przy realizacji zadań transportowych.

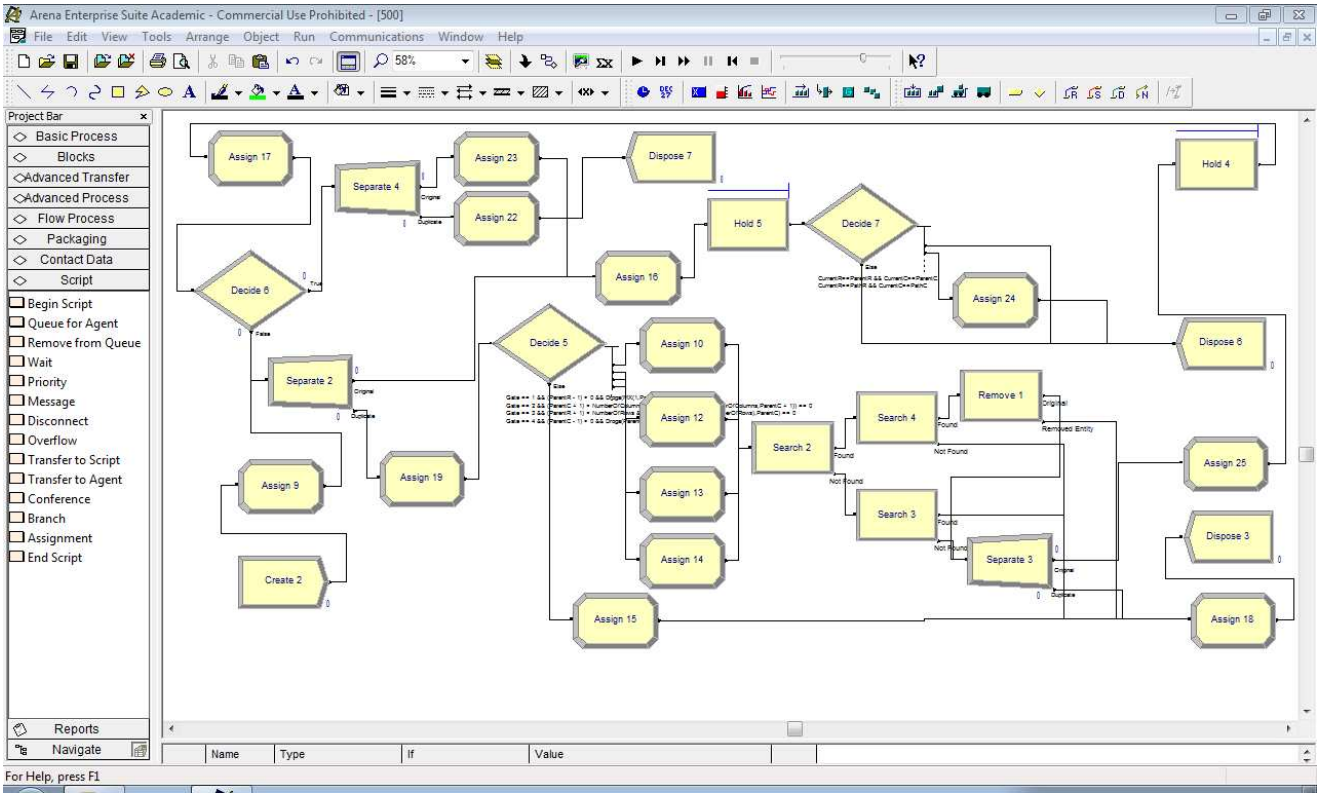
Na rysunku 1 pokazano przykład zdyskretyzowanego obszaru wraz z przeszkodami, który składa się z kwadratów ułożonych w sześciu wierszach i ośmiu kolumnach. Punkty początkowy i docelowy są środkami odpowiednich kwadratów. W klasycznym podejściu algorytm próbuje wyznaczyć najkrótszą drogę rozpatrując na każdym etapie wszystkie możliwe kierunki przemieszczania się do sąsiednich kwadratów. W sumie jest ich osiem. Zostało to pokazane na rysunku 1 za pomocą strzałek wychodzących z kwadratu zawierającego punkt początkowy. W prezentowanym artykule możliwe kierunki poszukiwań zostały ograniczone do czterech. Dwa w pionie i dwa w poziomie. Nie zmienia to istoty działania samego algorytmu a może znaleźć uzasadnienie praktyczne w przypadku założenia, że wszystkie drogi transportowe przecinają się pod kątem prostym. Dla przyjętych powyżej założeń zaimplementowano w programie Arena algorytm A* za pomocą bloków funkcjonalnych, rysunek 2. Następnie wykorzystano ten algorytm do wyznaczenia najkrótszej drogi z punku początkowego w kwadracie o współrzędnych (3,3) do docelowego o współrzędnych (3,7). Wymiary kwadratów zostały zdefiniowane jako dziesięć umownych jednostek. Tym samym odległość pomiędzy środkami kwadratów wynosiła 10 jednostek.



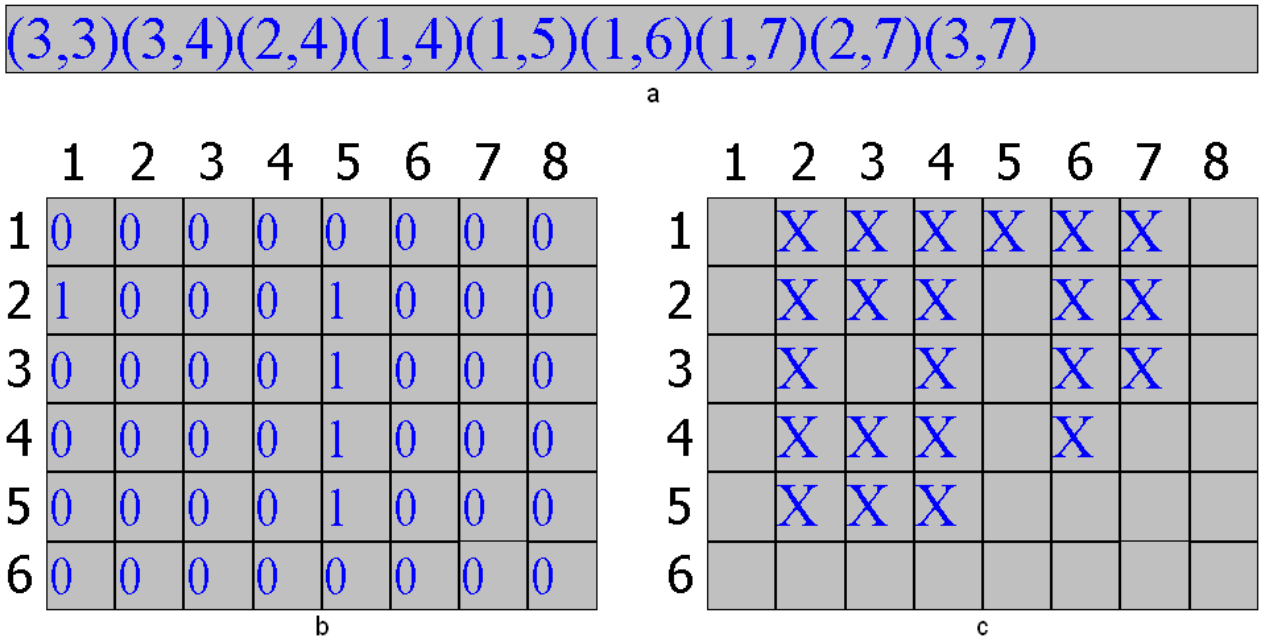
Rys.1. Przykład zdyskretyzowanego obszaru dróg transportowych

W efekcie działania algorytmu został wyznaczony ciąg współrzędnych kwadratów od początkowego do docelowego, tworzący trasę przejazdu, rysunek 3a. W celu zapisania informacji o tym, które kwadraty są zajęte przez przeszkody i muszą zostać ominięte, zdefiniowano tablicę o wymiarach 6 na 8. Za pomocą wartości „0” oznaczono kwadraty przejezdne a za pomocą wartości „1” zajęte przez przeszkody, rysunek 3b. W celu prezentacji metody działania algorytmu, na rysunku 3c za pomocą znaków „X” zaznaczono wszystkie kwadraty, które były analizowane przez algorytm podczas wyznaczania najkrótszej drogi. Jak można zauważyć, podczas wyznaczania drogi, algorytm na każdym etapie obliczeń sprawdza sąsiadujące kwadraty osiągalne we wszystkich kierunkach. Podczas definiowania dróg w podsystemach

transportowych zautomatyzowanych systemów produkcyjnych celowe jest określenie dopuszczalnych kierunków ruchu a tym samym dróg jedno i dwukierunkowych.



Rys.2. Implementacja algorytmu A w programie Arena za pomocą bloków funkcjonalnych*



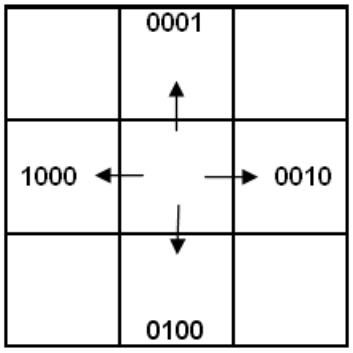
*Rys.3. Wynik działania algorytmu A**

Takie zawężenie jest często uzasadnione względami logistycznymi i dążeniem do minimalizacji powierzchni hal produkcyjnych a tym samym kosztów. Z tego względu w artykule zaproponowano metodę kodowania dopuszczalnych kierunków ruchu za pomocą informacji zapisanej na czterech bitach.

3.2 Kodowanie kierunków jazdy w postaci bitowej

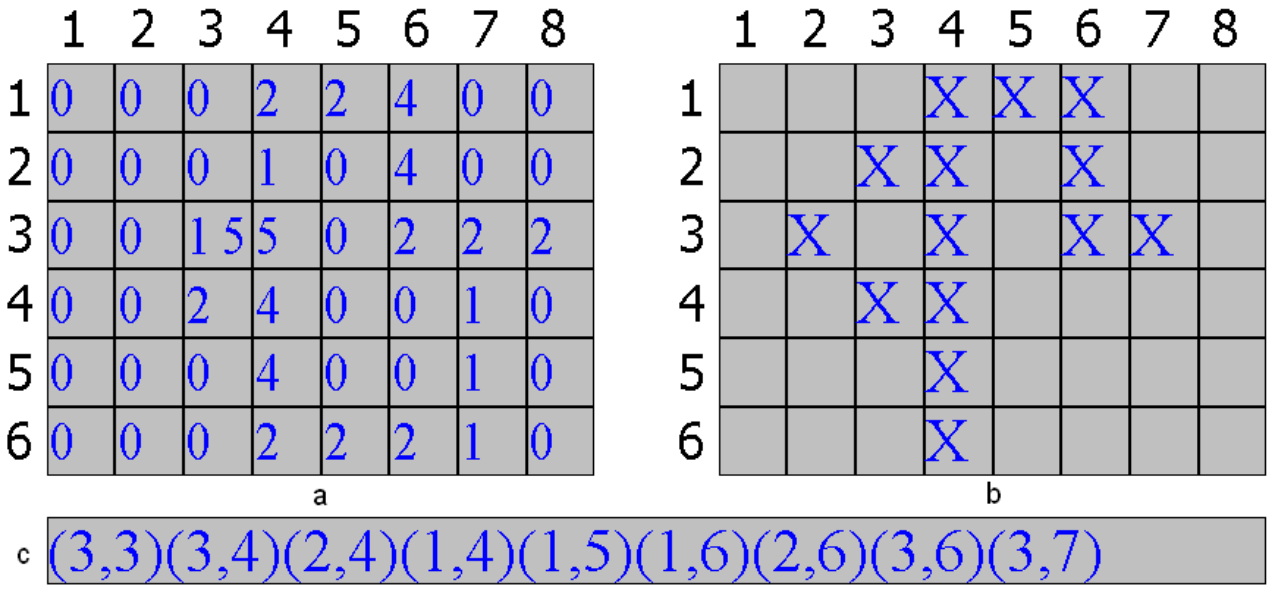
Do kodowania dopuszczalnych kierunków jazdy postanowiono wykorzystać już zdefiniowaną tablicę, która służyła do zapisu informacji o przejezdności poszczególnych komórek. W tablicy tej były wpisywane tylko wartości „0” lub „1”.

W celu kodowania do tablicy będą wpisywane też inne wartości, reprezentujące zakodowaną informację. Przyjęto następujący sposób kodowania. Z każdego kwadratu można przejść do sąsiedniego w czterech kierunkach, rysunek 4. Każdemu kierunkowi przypisano określoną pozycję bitu w słowie czterobitowym. Wartość „1" na pierwszym miejscu (od prawej) w słowie czterobitowym oznacza możliwość przejścia pionowo w górę. Możliwość przejścia poziomo w prawo jest reprezentowana przez wartość „1" na drugim miejscu itd. Zapis „0011" oznacza możliwość przejścia w górę i w prawo. W ten sposób można zdefiniować wszystkie możliwe warianty dopuszczalnych kierunków, w których można się przemieścić do sąsiedniego kwadratu. W celu zapisania tej informacji w tablicy jest ona przeliczana z postaci dwójkowej na dziesiętną. Np. dla słowa „**0011**" otrzymujemy 8***0** + 4***0** + 2***1** + 1***1** = 3.



Rys.4. Metoda kodowania kierunków

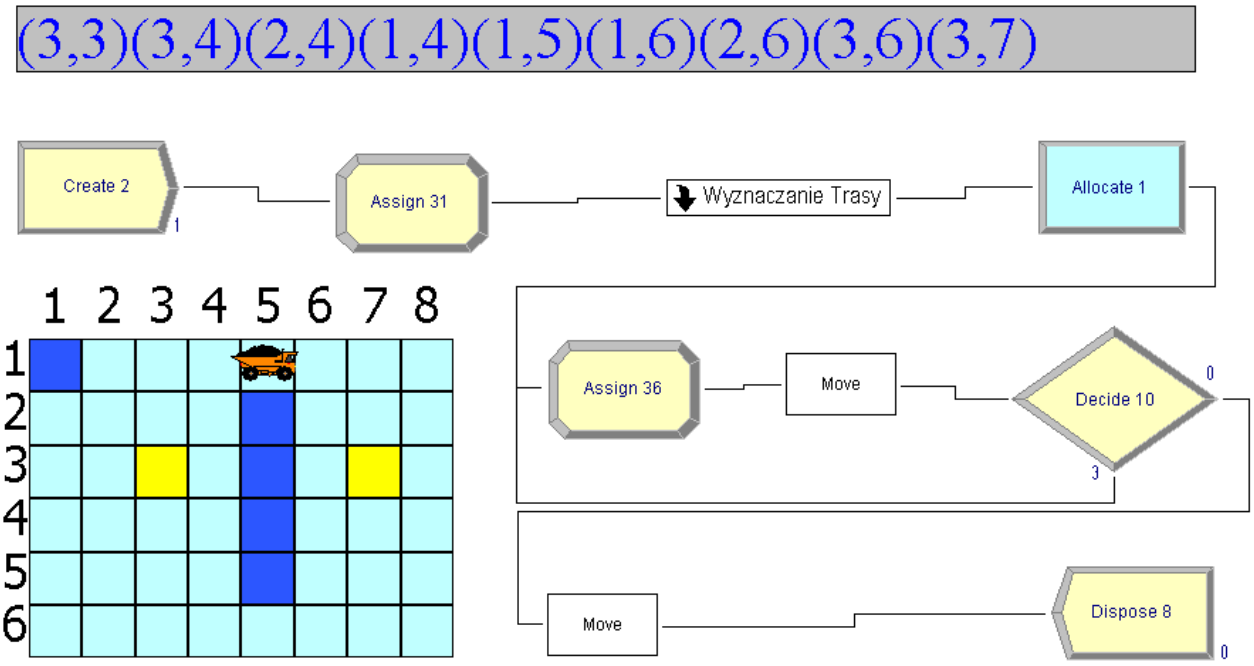
Dzięki takiemu sposobowi kodowania wystarczy wpisać do tablicy tylko jedną wartość liczbową i nie ma potrzeby tworzenia dodatkowej tablicy. Jednak takie rozwiązanie wymaga modyfikacji algorytmu. Konieczne jest bowiem dekodowanie informacji zapisanej w tablicy. W tym celu, za pomocą bloków funkcjonalnych, dokonano odpowiedniej modyfikacji algorytmu i przeprowadzono ponowne wyznaczanie najkrótszej drogi. Na rysunku 5a przedstawiono tablicę z przykładowo zakodowanymi informacjami. Na rysunku 5b za pomocą znaków „X" zaznaczono wszystkie kwadraty, które były analizowane przez algorytm podczas wyznaczania najkrótszej drogi w oparciu o zakodowaną informację. Na rysunku 5c pokazano wynik poszukiwań. Porównując rysunek 3c i 5b widać, że poszukiwania zostały ograniczone tylko do kwadratów dopuszczonych przez zakodowaną informację w tablicy. Takie ograniczenie ma istotny wpływ na szybkość obliczeń, szczególnie w przypadku obszarów składających się z dużej liczby kwadratów.



Rys.5. Wynik działania algorytmu w oparciu o kodowaną informację

Zaimplementowany algorytm działający w oparciu o zakodowaną informację został wykorzystany do sterowania przykładowym podsystemem transportowym w modelu przedstawionym na rysunku 6b. W module „Create 2" jest tworzona jednostka reprezentująca przedmiot, który ma zostać przewieziony. W module „Assign 31" jest przedmiotowi przypisywane zadanie transportowe (punkt początkowy i docelowy). W podmodelu „Wyznaczanie Trasy" zostaje wyznaczona trasa przejazdu bazując na zakodowanej informacji i wyświetlona w postaci ciągu współrzędnych kwadratów, tworzących trasę przejazdu, rysunek 6a. Po wyznaczeniu trasy w module „Allocate 1" jest przedmiotowi przydzielany

środek transportu (ciężarówka). W kolejnych modułach jest realizowane przemieszczanie przedmiotu za pomocą środka transportu z punktu początkowego do docelowego. Na rysunku 6c jest przedstawiona wizualizacja realizowanego zadania transportowego.



Rys.6. Model przykładowego podsystemu transportowego

4. WNIOSKI

Implementacja algorytmu A* za pomocą bloków funkcjonalnych do programu Arena pozwoliła na swobodne wyznaczanie najkrótszych tras przejazdu a tym samym na przejęcie pełnej kontroli nad realizacją zadań transportowych. Dzięki temu możliwe jest budowanie modeli różnych podsystemów transportowych i testowanie nowych metod sterowania nimi. Zastosowanie kodowania dopuszczalnych kierunków jazdy za pomocą informacji zapisanej w postaci czterech bitów w znaczący sposób ułatwia budowanie modeli symulacyjnych o raz działanie samego algorytmu A*. Dzięki temu rozwiązaniu bardzo łatwe staje się modelowanie dróg transportowych jedno i dwukierunkowych. Rozszerzenie kodowania do zapisu na ośmiu bitach pozwoli na kodowanie ośmiu kierunków jazdy (w tym na skos). Przeprowadzona symulacja uproszczonego modelu sterowania ruchem pojazdu podczas realizacji zadania transportowego w pełni wykazała przydatność zaproponowanego rozwiązania.

5. BIBLIOGRAFIA

[1] Dechter R, Perl J.: *Generalized best-first search strategies and the optimality of A**, Journal of the ACM 32 (3) (1985), pp. 505–536.
[2] Ebendt R., Drechsler R.: Weighted A* search – unifying view and application, Artificial Intelligence 173 (2009) 1310–1342
[3] Kelton W. D., Sadowski R. P., Sturrock D. T.: *Simulation with Arena*, McGraw-Hill, New York 2004
[4] Koenig S. et al.: *Lifelong Planning A**, Artificial Intelligence 155 (2004) 93–146
[5] Le-Anh T., De Koster M.B.M.L: *A review of design and control of automated guided vehicle systems*, European Journal of Operational Research 171 (2006), pp.1–23.
[6] Lester P.: *A* Pathfinding for Beginners*, http://www.policyalmanac.org/games/aStarTutorial.htm
[7] Likhachev M. et al.: *Anytime search in dynamic graphs*, Artificial Intelligence 172 (2008) 1613–1643
[8] Zając J., Krupa K., Słota A., Więk T.: *Autonomiczna platforma mobilna do realizacji transportu międzyoperacyjnego – projekt wstępny*, LOGISTYKA, Nr 6, 2010, s.3779-3788.
[9] Zając J, Słota A., Chwajoł G.: *Distributed Manufacturing Control: Models and Software Implementations. Management and Production Engineering Review*, Vol. 1., No. 1., May 2010, s. 38-56.