mgr inż. Patryk Widuliński
Wydział Elektroniki i Informatyki
Politechnika Koszalińska

Algorytmy detekcji infekcji programów komputerowych inspirowane biologicznymi mechanizmami immunologicznymi

Rozprawa doktorska
napisana pod kierunkiem
prof. dra hab. inż.
Krzysztofa Wawryna

Koszalin 2022

Podziękowania

Pragnę podziękować mojemu promotorowi,
prof. dr hab. inż. Krzysztofowi Wawrynowi
za ofiarowany mi czas, za zaangażowanie
i merytoryczną pomoc podczas pisania rozprawy doktorskiej.

Oświadczenie

Ja, Patryk WIDULIŃSKI, zaświadczam, że niniejsza dysertacja o tytule "Algorytmy detekcji infekcji programów komputerowych inspirowane biologicznymi mechanizmami immunologicznymi" i praca w niej zawarta jest moją własnością. Potwierdzam, że:

- Ta praca została wykonana w całości lub głównie w trakcie ubiegania się o stopień naukowy na Politechnice Koszalińskiej.
- W przypadku, gdy jakakolwiek część tej pracy została wcześniej złożona w celu uzyskania stopnia naukowego lub jakichkolwiek innych kwalifikacji na tej uczelni lub jakiejkolwiek innej instytucji, zostało to wyraźnie stwierdzone.
- Wszędzie tam, gdzie odnoszę się do opublikowanych prac innych, zawsze jest to wyraźnie oznaczone.
- Wszystkie cytaty pochodzące z innych prac, są wyraźnie oznaczone i spisane w bibliografii. Wyjątkiem jest przedstawiona w niniejszej pracy teza, która jest całkowicie moją własną pracą.
- Uznałem wszystkie główne źródła pomocy.
- Tam, gdzie teza opiera się na pracy wykonanej przeze mnie wspólnie z innymi, wyjaśniłem dokładnie to, co zostało zrobione przez innych i co sam wniosłem.

Podpis: _____

Data: _____

Streszczenie

Systemy komputerowe są w dzisiejszych czasach bardzo popularne. Niestety, wraz ze wzrostem ich popularności pojawili się aktorzy, którzy w różnych celach dążą często do nieuprawnionego dostępu do danych użytkowników. W celu znalezienia odpowiedzi na zagrożenia powstały IDS (ang. intrusion detection systems) - systemy wykrywania intruzji. W ostatnich latach naukowcy zainspirowani byli do znalezienia nowoczesnych rozwiązań pozwalających na wykrywanie intruzji. Badacze zauważyli pewne paralele między celami IDS i biologicznych systemów immunologicznych. Dzięki temu powstały sztuczne systemy immunologiczne (AIS - ang. artificial immune systems). Celem rozprawy była analiza istniejących i opracowanie własnego algorytmu detekcji infekcji programów komputerowych, inspirowanego mechanizmami obronnymi organizmów żywych. Rozprawa stawia tezę, że możliwa jest realizacja algorytmów detekcji infekcji programów komputerowych o właściwościach porównywalnych lub lepszych od znanych rozwiązań, w tym znanych z literatury rozwiązań wykorzystujących AIS. W niniejszej rozprawie doktorskiej przeprowadzono kompleksowy przegląd literatury dotyczącej IDS i AIS, i w szczególności najpopularniejszego algorytmu AIS, jakim jest algorytm negatywnej selekcji (NSA - ang. negative selection algorithm). Zaproponowano następnie własny system wykrywania intruzji, który pozwala na wykrywanie infekcji w systemie operacyjnym. W systemie zaimplementowano metodę losową i metodę wzorców, a następnie przedstawiono

autorskie ulepszenie tych metod przy pomocy dodatkowego zbioru receptorów międzykomórkowych. System dogłębnie przetestowano dla szerokiej gamy parametrów wejściowych generacji receptorów, a następnie porównano osiągi z innymi rozwiązaniami znanymi z literatury. Badania eksperymentalne pokazały, że system IDS osiąga wyniki porównywalne lub lepsze od wyników znanych z literatury. Na podstawie badań eksperymentalnych wyciągnięto wnioski, że cel pracy został osiągnięty, a teza dowiedziona. Dalsze badania mogą uwzględniać możliwości naprawy zainfekowanych plików w systemie operacyjnym.

Abstract

Computer systems are very popular these days. Unfortunately, with their increased popularity has come the emergence of actors who, for different purposes, often seek unauthorized access to user data. IDS (intrusion detection systems) have been developed to address these threats. In recent years, researchers have been inspired to find modern solutions to detect intrusions. Researchers have noticed some parallels between the goals of IDS and biological immune systems. This has led to the development of artificial immune systems (AIS). The aim of the thesis was to analyze existing and develop my own algorithm for detection of computer program infections, inspired by defense mechanisms of living organisms. The thesis states that it is possible to implement algorithms for detection of computer program infections with properties comparable or better than known solutions, including solutions using AIS known from the literature. In this dissertation, a comprehensive review of the literature on IDS and AIS, and in particular the most popular AIS algorithm, the negative selection algorithm (NSA), has been conducted. A custom intrusion detection system was then proposed to detect infections in the operating system. A random method and a template method were implemented in the system, and a proprietary improvement of these methods using an additional set of intercellular receptors was presented. The system was thoroughly tested for a wide range of receptor generation input parameters, and performance was compared with other solutions known

from the literature. The experimental studies showed that the IDS system achieves comparable or better results than those known from the literature. Based on the experimental studies, it was concluded that the goal of the thesis was achieved and the thesis was proven. Further research may consider the ability of the IDS to repair infected files in the operating system.

Spis treści

Rozdział 1

Wprowadzenie

Systemy komputerowe są w dzisiejszych czasach bardzo popularne. Niestety, wraz ze wzrostem ich popularności pojawili się aktorzy, którzy w różnych celach dążą często do nieuprawnionego dostępu do danych użytkowników. Systemy komputerowe są zagrożone nie tylko przez ludzi, którzy pragną sieciowo operować na przejętych stacjach roboczych, ale także przez oprogramowanie typu malware. W celu znalezienia odpowiedzi na zagrożenia powstały IDS (ang. intrusion detection systems) - systemy wykrywania intruzji. IDS służą do wykrywania i zwalczania infekcji sieciowych i lokalnych. Stanowią one od kilkudziesięciu lat ważną i mocno rozwijaną klasę oprogramowania. IDS stosują różne metody w celu mitygacji zagrożeń. Takimi metodami mogą być na przykład filtrowanie pakietów sieciowych według reguł, lub baza danych sygnatur programów antywirusowych. Metody te często nie pozwalają na wykrywanie nieznanych wcześniej zagrożeń. Pierwsze systemy IDS inspirowane sztucznymi mechanizmami immunologicznymi (AIS - ang. artificial immune systems) powstawały w celu przezwyciężenia ograniczeń tych poprzednich. IDS korzystające z AIS opierają się zwykle na algorytmie selekcji negatywnej (NSA - ang. negative selection algorithm) [36], pozytywnej [2] i klonalnej [60]. Są to algorytmy inspirowane biologicznymi systemami

immunologicznymi, a podejście NSA osiągnęło największe zainteresowanie wśród naukowców na świecie. Algorytm selekcji negatywej zakłada generację zbioru receptorów, które stanowią odpowiednik przeciwciał i limfocytów typu T w biologicznym układzie odpornościowym.

Receptory mogą być generowane na różne sposoby: losowo [26], przy pomocy wzorców [37] i inne. Receptory oparte są również o parametr zwany progiem aktywacji, który w zależności od podejścia, może być stały [47] lub zmienny [42]. Zastosowanie tych rozwiązań często było ograniczone np. wielkością przetwarzanych plików czy istnieniem dziur powodujących duży procent niewykrywalnych infekcji. W celu przezwyciężenia tych ograniczeń w ostatnich kilkunastu latach powstało wiele rozwiązań modyfikujących NSA wykorzystujących różne udoskonalone metody uczenia (trenowania) zbioru receptorów za pomocą np. liczb rzeczywistych (real-values) [20], diagramu Voronoia [89], dwuetapowego trenowania [28], hierarchicznego grupowania [7], algorytmu genetycznego [27] czy mechanizmów immunoregulacji adaptacyjnej [18]. W rozwiązaniach tych poszukiwane są optymalne sposoby trenowania receptorów. Przeważają w nich podejścia ze zmiennymi programi aktywacji. Dają one coraz lepsze możliwości szybkiego przetwarzania dużych plików czy eliminacji dziur. Wstępne badania z dwoma progami aktywacji dla dwóch różnych zbiorów receptorów [51, 71, 72, 73, 74, 75, 76] pokazały, że otrzymane rozwiązania dają obiecujące rezultaty. Jest to podstawa do stwierdzenia, że istnieje możliwość otrzymania wysokiej jakości systemu detekcji infekcji - IDS.

Celem niniejszej pracy jest więc analiza istniejących i opracowanie własnego algorytmu detekcji infekcji programów komputerowych, inspirowanego mechanizmami obronnymi organizmów żywych.

Rozprawa stawia tezę, że możliwa jest realizacja algorytmów detekcji infekcji programów komputerowych o właściwościach porównywalnych lub lepszych

od znanych rozwiązań, w tym znanych z literatury rozwiązań wykorzystujących sztuczne systemy immunologiczne.

W rozdziale 2 omówiono niezbędne do zrozumienia istoty rozprawy pojęcia fundamentalne i przeprowadzono kompleksowy przegląd dostępnej w środowisku naukowym literatury dotyczącej omawianych zagadnień. Rozdział 3 zawiera szczegółowy opis proponowanego w niniejszej rozprawie rozwiązania uwzględniający między innymi część teoretyczną i schematy blokowe, ilustrujące zaimplementowane w systemie algorytmy. W rozdziale 4 opisana została metodyka i przedstawione zostały wyniki badań przeprowadzonych z wykorzystaniem systemu IDS. Porównano następnie wyniki implementacji systemu z wynikami osiągniętymi przez implementacje rozwiązań proponowanych w literaturze. Pracę podsumowano i wyciągnięto wnioski w rozdziale 5.

Rozdział 2

Pojęcia podstawowe i przegląd literatury

2.1 Bezpieczeństwo systemu komputerowego

Pojęciem nadrzędnym, które należy zdefiniować, jest bezpieczeństwo systemu komputerowego. Jednak by to zrobić, należy wprowadzić najpierw pojęcie systemu komputerowego. Systemem komputerowym (SK) nazywamy określoną kombinację układów umożliwiających użytkownikom realizację konkretnych zadań obliczeniowych [22]. Instancję systemu komputerowego nazywamy stacją roboczą. Elementy systemu komputerowego można podzielić na fizyczne i logiczne [22]. Do elementów fizycznych można zaliczyć sprzęt komputerowy taki jak płyta główna, pamięć operacyjna, procesor czy dysk twardy [55]. Do elementów logicznych zaliczają się różnego rodzaju dane: konfiguracja komponentów fizycznych (na przykład ustawienia UEFI/BIOS), oprogramowanie układów (firmware), system operacyjny nadzorujący pracę komputera i dane użytkowników na dysku twardym [22, 55].

Bezpieczeństwo systemu komputerowego (BSK) jest jego stanem odporności

na zagrożenia [62]. Zapewnienie bezpieczeństwa jest wysiłkiem mającym zaowocować ochroną sprzętu i danych stacji roboczej [62]. Mówiąc o bezpieczeństwie systemu komputerowego, można mówić więc o bezpieczeństwie elementów fizycznych i o bezpieczeństwie danych tego systemu. Bezpieczeństwo elementów fizycznych zapewniane jest przez zabezpieczenie danej stacji roboczej przed fizycznym niepowołanym dostępem osób trzecich. Bezpieczeństwo danych natomiast zapewniane jest przez konfigurację UEFI/BIOS, oprogramowanie firmware i system operacyjny.

Można wyróżnić kilka podstawowych aspektów [37] wchodzących w skład BSK:

- Dostępność – system komputerowy powinien zapewniać możliwie nieprzerwany dostęp do swoich zasobów i danych użytkownikom, którzy mają do tego uprawnienia [46],
- Poufność danych – SK powinien zapewniać poufność danych użytkowników i blokować dostęp do danych użytkownikom niepowołanym [21],
- Integralność – dane w SK powinny być chronione przed niepożądanymi modyfikacjami (usunięcie/nadpisanie/uszkodzenie) [35],
- Poprawność klasyfikacji zagrożeń – oprogramowanie zabezpieczające SK powinno dążyć do minimalizacji zdarzeń wykrycia fałszywie pozytywnych [23],
- Odpowiedzialność za dane – system komputerowy powinien posiadać wbudowany system logowania informacji tak, by w sytuacji naruszenia bezpieczeństwa możliwe było wykrycie takiej sytuacji i jej ewentualnych sprawców [80].

Zależnie od pewnych czynników, istotne może być skupienie się na określonych aspektach bezpieczeństwa z powyżej wymienionych. Na przykład, jeżeli stacja

robocza służy do archiwizacji danych, ważne może być w tym przypadku skoncentrowanie się na aspekcie integralności danych w systemie. Politykę bezpieczeństwa (ang. security policy) systemu komputerowego stanowi wybór aspektów bezpieczeństwa, na których skupia się administrator systemu [52].

Intruzją (lub włamaniem) nazywamy czyn dostępu do danych systemu komputerowego przez niepowołane osoby trzecie. Mówiąc o dostępie do danych należy zaznaczyć, że chodzi tu nie tylko o odczyt danych, ale także ich modyfikacja lub usunięcie. W takim przypadku następuje naruszenie zasad dostępności, poufności danych i integralności bezpieczeństwa systemu komputerowego. Intruzje mogą być popełniane bezpośrednio przez aktorów ludzkich (na przykład przez hakerów) i automatycznych (przy pomocy oprogramowania typu malware [67]). Włamania dokonywane przez szkodliwe oprogramowanie mają często charakter włamań wstępnych, torujących drogę aktorom ludzkim do niepowołanego dostępu do danych [12].

Realizacja obranej polityki bezpieczeństwa systemu opiera się na doborze odpowiednich metod do rozwiązywanych problemów. W celu ochrony stacji roboczej przed włamaniami administrator może wykorzystać oprogramowanie specjalizujące się w zapobieganiu tego typu działaniom. I tak, zabezpieczanie systemu przed sieciowymi próbami włamania może rozpocząć się instalacją oprogramowania zapory ogniowej (ang. firewall), dzięki któremu administrator może między innymi zablokować wybrane porty sieciowe systemu. Blokada tego rodzaju znacznie utrudnia w takiej sytuacji atak na określonych przez administratora portach [52, 82].

Kolejnym ważnym etapem zabezpieczania systemu jest instalacja oprogramowania wykrywającego szkodliwe oprogramowanie.

2.2 Systemy wykrywania intruzji

Na przestrzeni ostatnich lat wśród naukowców spopularyzowały się narzędzia zwane systemami wykrywania intruzji (ang. intrusion detection systems – IDS) [9, 58, 63, 66, 68, 70, 75, 76]. Narzędzia te mają za zadanie odróżnić zdarzenia pożądane od niepożądanych przy pomocy określonych metod działania. IDS wykorzystywane są głównie do wykrywania niepożądanych działań sieciowych, ale mogą być także używane detekcji zagrożeń lokalnych [17]. Podstawowymi technikami stosowanymi w IDS są wykrywanie oparte na regułach i wykrywanie oparte na profilach [63]. Pierwsza z technik opiera się na dopasowywaniu ciągu próbek do znanych już wcześniej wzorców, o których wiadomo, że są szkodliwe. Wzorce takie nazywamy w takim przypadku sygnaturami. Druga z wymienionych technik polega na analizie zachowania systemu i wykrywania zachowań niezgodnych z „normalnym" działaniem środowiska.

Podstawowe techniki IDS nie uwzględniają dynamicznego uczenia się i adaptacji na podstawie zmitygowanych już intruzji i nie pozwalają na zaawansowane wykrywanie nieznanych zagrożeń. W związku z tym naukowcy zainspirowani byli do poszukiwania nowoczesnych rozwiązań komponujących się z tematyką IDS.

Jednym z rozwiązań znanych z natury jest biologiczny system immunologiczny (ang. biological immune system – BIS). Układ ten zbudowany jest ze struktur biologicznych i procesów w organizmie, które chronią organizm przed chorobami. W celu poprawnego funkcjonowania system immunologiczny musi mieć zdolność wykrywania szerokiej gamy przedstawicieli gatunków szkodliwych, takich jak wirusy, bakterie i pasożyty, i odróżniać je od zdrowej tkanki własnego organizmu [37]. Podstawowymi elementami układu immunologicznego są układ wrodzony i układ adaptacyjny.

Układ wrodzony, występujący u prawie wszystkich organizmów żywych, daje organizmom odporność wrodzoną (nieswoistą). Odpowiedź tego układu na eks-

pozycję szkodnika jest natychmiastowa. Sama ekspozycja nie jest zapamiętywana przez układ, który nie buduje pamięci immunologicznej [37].

Układ adaptacyjny pozwala organizmowi na budowanie pamięci immunologicznej i daje mu odporność swoistą. Komórkami, które między innymi biorą udział w tym procesie są limfocyty typu T i limfocyty typu B. Limfocyty typu T odpowiedzialne są za odpowiedź komórkową, eliminującą zainfekowane lub zmutowane komórki, a limfocyty typu B odpowiedzialne są za wytwarzanie białek zwanych przeciwciałami. Przeciwciała wiążą antygeny na powierzchniach szkodliwych komórek „oznaczając" je, co pozwala na uruchomienie odpowiedzi humoralnej [37, 53].

2.3 Sztuczne systemy immunologiczne

Dzięki podobieństwom w zastosowaniach systemów wykrywania intruzów i układów immunologicznych powstały sztuczne systemy immunologiczne (ang. artificial immune systems – AIS) [25, 38, 77]. AIS stanowią zbiór metod obliczeniowych inspirowanych biologicznymi układami immunologicznymi. Atrakcyjność AIS wynika z ich zdolności do uczenia się i adaptacji w środowisku [25, 85]. Najważniejszą cechą systemów IDS opartych na algorytmach AIS jest umiejętność rozróżniania „komórek" własnych od obcych. Prominentnymi algorytmami stosowanymi w zagadnieniach związanych z AIS są między innymi algorytm selekcji negatywnej [17], algorytm selekcji pozytywnej [87] i algorytm selekcji klonalnej [60]. Algorytmy AIS są w pewnych aspektach podobne do sieci neuronowych, ponieważ uwzględniają trenowanie systemu na podstawie określonego zestawu danych [14, 54].

Algorytm selekcji negatywnej (ang. negative selection algorithm – NSA) inspirowany jest układem adaptacyjnym biologicznego systemu immunologicznego.

Jego działanie polega na generacji ciągów binarnych, które potrafią dopasowywać ciągi obce, a jednocześnie nie dopasowują nigdy ciągów własnych [3, 41]. Jeżeli wygenerowany ciąg binarny dopasowuje ciąg własny, jest eliminowany. Działanie to jest analogiczne do działania układu adaptacyjnego, który generuje przeciwciała wiążące się tylko ze szkodliwymi antygenami i limfocyty typu T rozpoznające tylko obce komórki [13].

Algorytm selekcji pozytywnej (ang. positive selection algorithm – PSA) działa podobnie jak NSA, z tym, że zamiast dopasowywać ciągi obce, wygenerowane przez algorytm ciągi dopasowują tylko ciągi własne [2, 87].

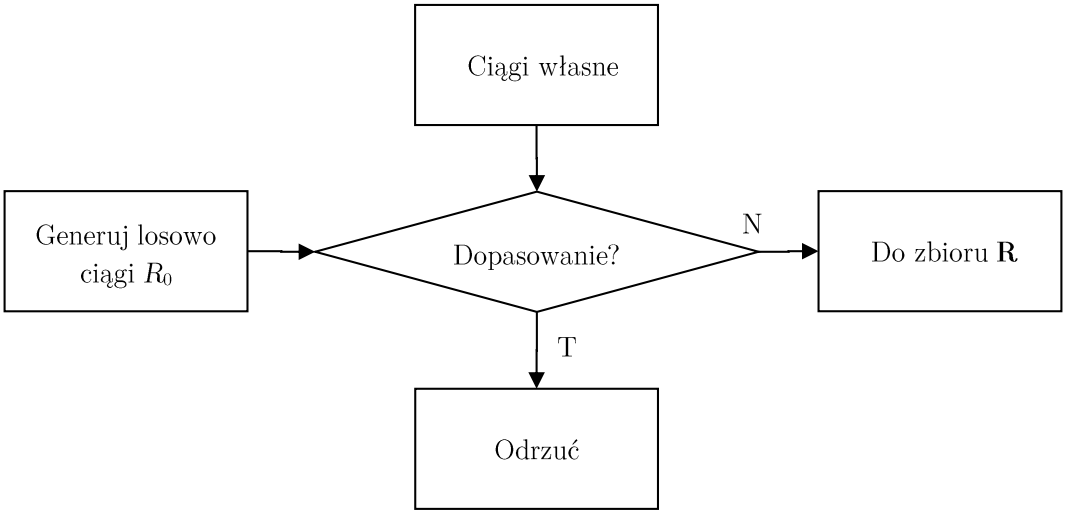
Algorytm selekcji klonalnej (ang. clonal selection algorithm – CSA) – działanie tego algorytmu inspirowane jest reakcją biologicznego systemu immunologicznego, powodującą namnożenie przeciwciał wykrywających dany antygen. Aktywacja limfocytów typu B (dla konkretnych przeciwciał) stymuluje ich klonowanie, a następnie intensywną mutację genetyczną przeciwciał w celu poprawienia ich dopasowania do antygenu [16]. Analogicznie, algorytm wykrywa najlepiej dopasowane ciągi binarne i klonuje je w celu dalszej mutacji i poprawy stopnia dopasowania zmutowanych ciągów [15]. Jest stosowany jako algorytm pomocniczy do NSA i PSA.

Algorytmy NSA i PSA mają swoje zalety i wady. W pracy [69] pokazano, że wydajność wykrywania jest lepsza w przypadku zastosowania algorytmu pozytywnej selekcji. Jednak w przypadku gdy liczba ciągów wygenerowanych przez NSA jest mniejsza od liczby ciągów własnych, to algorytm negatywnej selekcji może okazać się lepszy [42].

W związku z naturą podmiotowych ciągów własnych, do dalszych rozważań w niniejszej rozprawie wybrany został algorytm selekcji negatywnej.

2.4 Algorytm negatywnej selekcji

Celem działania algorytmu negatywnej selekcji (NSA) w swojej podstawowej wersji jest utworzenie zbioru ciągów zdolnych do wykrycia intruzów [47]. Na rysunku 1 pokazano schemat ogólny działania NSA, bazowany na [26].



Rysunek 1. Schemat ogólny algorytmu negatywnej selekcji

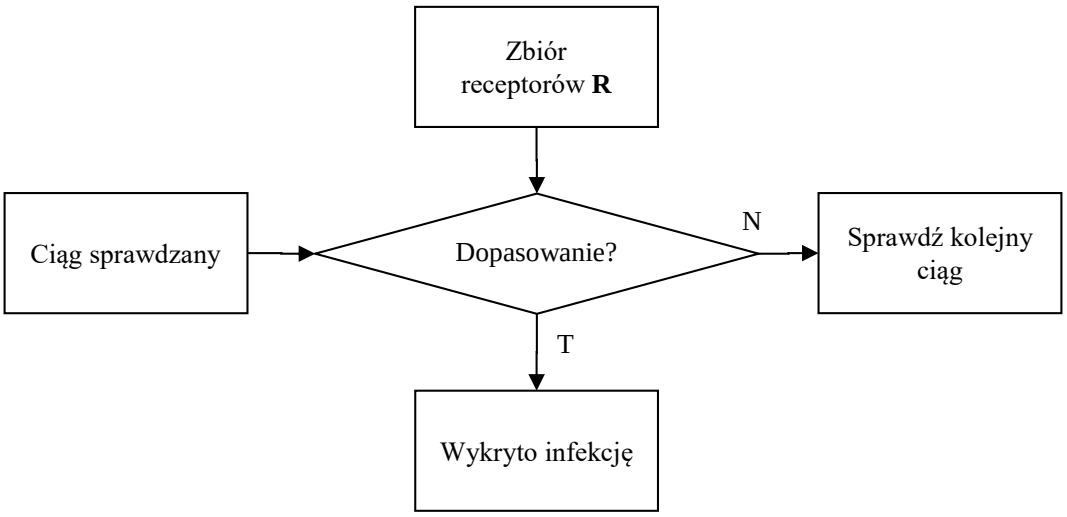
Ciągi generowane przez algorytm nazywane są receptorami lub detektorami. Każdy receptor ma określoną długość oznaczoną przez l [26]. Każdy kandydat na receptor sprawdzany jest pod kątem dopasowania do dowolnego ciągu własnego, przy czym przez pojęcie ciągu własnego rozumiemy ciąg, który nigdy nie powinien być wykryty jako anomalia. W tym celu NSA przyjmuje parametr m stanowiący próg aktywacji receptora [31]. Mówimy, że receptor został aktywowany, jeżeli dopasowuje inny ciąg binarny. Dopasowanie polega na wystąpieniu zarówno w receptorze, jak i w sprawdzanym ciągu binarnym m identycznych kolejnych bitów na jednakowej pozycji k [37].

Jeżeli wygenerowany ciąg dopasowuje przynajmniej jeden ciąg własny, to nie może zostać receptorem i jest eliminowany. Klasycznie, NSA zakłada istnie-

nie jednego zbioru receptorów **R**, który zawiera wszystkie wygenerowane receptory [26].

Metoda generacji receptorów nie jest narzucona – dla ustalenia uwagi omówiona zostanie metoda generacji losowej [26]. Do generacji losowej niezbędny jest parametr R_{max} stanowiący maksymalną liczbę receptorów do wygenerowania. Dla zadanego parametru l generowane jest R_{max} kandydatów na receptory. Każdy kandydat na receptor podlega opisanej wyżej weryfikacji, po czym umieszczony jest w wynikowym zbiorze **R**. Receptory wygenerowane przez NSA z reguły nie pozwalają na uzyskanie 100% wykrywalności anomalii. Obszary, które nie zostały pokryte przez receptory nazywamy dziurami [19].

Algorytm negatywnej selekcji zastosowany może zostać również jako podstawa mechanizmu detekcji infekcji w IDS [10]. Na rysunku 2 przedstawiono ogólny schemat przykładowej adaptacji NSA do zadania wykrywania infekcji w systemie.



Rysunek 2. Schemat ogólny algorytmu negatywnej selekcji wykorzystanego do detekcji infekcji

W przypadku zastosowania NSA do detekcji infekcji, wejście z generatora receptorów zastąpione jest strumieniem ciągów do sprawdzenia przez IDS. Zbiór

ciagów własnych zostaje zastąpiony zbiorem receptorów **R**. Dopasowanie sprawdzane jest dla tych samych parametrów l i m co w przypadku generacji receptorów. W przypadku dopasowania chociaż jednego ciągu własnego algorytm kończy pracę, informując o wykryciu infekcji, co jest kolejną różnicą w stosunku do wersji z generacją receptorów (wtedy algorytm odrzucał kandydata na receptor).

Istotnymi wskaźnikami wydajności systemu IDS i algorytmów w nim zastosowanych, w tym algorytmu negatywnej selekcji, są:

- *TP* (ang. true positives) – liczba poprawnie wykrytych infekcji,
- *TN* (ang. true negatives) – liczba poprawnie niewykrytych infekcji,
- *FP* (ang. false positives) – liczba niepoprawnie wykrytych infekcji,
- *FN* (ang. false negatives) – liczba niepoprawnie niewykrytych infekcji.

Pozostałymi wskaźnikami są:

- czas generacji receptorów,
- liczba receptorów w pamięci po generacji,
- zajętość pamięci przez receptory podstawowe,
- zajętość pamięci przez wszystkie receptory,
- zajętość pamięci przez oryginalny program.

W pracy znajdują się także inne warianty powyższych wskaźników, takie jak TP_I i FN_I. Zostaną one opisane w rozdziale 4.

2.5 Przegląd literatury i udoskonalonych algorytmów

Algorytmy sztucznych systemów immunologicznych są chętnie wykorzystywane, badane i udoskonalane w środowisku naukowym [36, 48, 57, 59, 86, 88]. González, Dasgupta, Kozma [29] wykorzystali do badań algorytmu reprezentację danych przy pomocy dwuwymiarowej płaszczyzny i liczb rzeczywistych dla przestrzeni ciągów własnych i obcych (RNSA – real-valued negative selection algorithm). Badania zostały przeprowadzone także dla reprezentacji liczb binarnych w kodzie Graya. Istotnym wnioskiem wyciągniętym przez autorów było stwierdzenie, że bardzo ważne jest adekwatne dobranie reguły dopasowania algorytmu negatywnej selekcji, zgodnie z przeznaczeniem IDS. Zaznaczono, że dla zastosowań w których znana jest cała przestrzeń ciągów własnych (takich jak na przykład skanowanie w celu sprawdzenia integralności danych), uogólnienie danych własnych nie jest aż tak istotne. Podobne rozwiązania oparte na reprezentacji przy pomocy liczb rzeczywistych przedstawiono w pracach [20, 30, 49, 84]. Ji, Dasgupta [44] napisali o problemach w zastosowaniu NSA opartego o liczby rzeczywiste. Podejście opisane w [4] zakłada wykorzystanie wielokształtnych receptorów w celu osiągnięcia lepszego pokrycia obszaru ciągów obcych. Ulepszone algorytmy negatywnej selekcji nadają się także do klasyfikacji wiadomości e-mail jako niechcianych (spam) [1, 8, 39, 40, 64].

Ji, Dasgupta [42] udoskonalili NSA przy pomocy detektorów o zmiennej długości (V-detektory). Detektory te, dzięki swojej zmiennej długości, lepiej "zatykają" dziury powstałe przy generacji. Badania wykazały, że wydajność algorytmu poprawiła się bez dużego wzrostu jego złożoności. Lu, Zhang, Wang, Gong [50] zaproponowali algorytm NSA oparty o V-detektory do wykrywania oprogramowania ransomware. W pracy [7] zaprezentowano szybki algorytm negatywnej selekcji

oparty na strukturze hierarchicznej zbioru ciągów własnych. Zhu, Chen, Yang, Li, Yang, Zhang [90] wykorzystali diagramy Voronoia w celu ulepszenia NSA. Zaproponowany przez autorów algorytm VorNSA konstruuje diagram Voronoia bazując na zbiorze testowym, a następnie generuje dwa rodzaje receptorów na podstawie tego diagramu, zmniejszając czas na generację receptorów. Etap testowania (detekcji) również został przeprojektowany – dane dzielone są na mniejsze przedziały, mapowane i sortowane na etapie redukcji. Innym podejściem wykorzystującym diagramy Voronoia jest podejście opisane w [89]. González, Dasgupta, Niño [32] przedstawili wersję algorytmu selekcji negatywnej, która została rozbudowana o szacowanie optymalnej liczby receptorów potrzebnych do pokrycia przestrzeni ciągów obcych (RRNSA – randomized real-value negative selection algorithm). Oprócz rozbudowania samego algorytmu autorzy przeprowadzili dogłębną analizę teoretyczną stanowiącą podstawy do analizy wydajności poprawionej przez nich wersji. Autorzy wywnioskowali, że wariant RRNS działa szybciej niż RNS, ale zaznaczyli jednocześnie, że w pewnych wypadkach algorytmy heurystyczne są jeszcze wydajniejsze, mimo że inne algorytmy mogą mieć lepszą bazę teoretyczną.

Marciniak, Wawryn, Widuliński [51] przedstawili wykorzystanie algorytmu negatywnej selekcji do sterowania kotłem ciepłowniczym. W [28] zawarto wersję algorytmu trenowanego kilkukrotnie dla innej liczby ciągów własnych w celu zwiększenia wydajności. Podejście zaproponowane w [18] uwzględnia zastosowanie mechanizmu adaptacyjnej immunoregulacji do obliczenia promienia na płaszczyźnie ciągów własnych. Saurabh, Verma [65] zaproponowali wersję algorytmu NSA z funkcją dostrajania o nazwie NIIAD. Balicki [5] przedstawił NSA do pokonania ograniczeń wielokryterialnego algorytmu ewolucyjnego. Prace [6, 81] pokazały, że AIS mogą być stosowane do wykrywania zagrożeń w mobilnych systemach operacyjnych. W [11] zaproponowany został system MILA (multilevel immune learning algoritm), który uwzględnia nie tylko zastosowanie NSA, ale także eks-

pansję receptorów przy pomocy metody klonalnej i dynamiczną metodę generacji receptorów w jednym rozwiązaniu. Fakhari, Moghadam [24] przedstawili wersję NSA o nazwie NSSAC, który ma zdolność dopasowywania się do zbiorów danych. Gao, Ovaska, Wang [27] zaproponowali w swojej pracy metodę generacji receptorów opartą o algorytm genetyczny. Praca [34] zawiera opis systemu IDS opartego na algorytmie ewolucyjnym w celu wykrywania anomalii w rozproszonych systemach komputerowych. W [43] zawarto informacje na temat szacowania zasięgu receptorów w NSA. Kamal, Bhusry [45] zaprezentowali algorytmy negatywnej selekcji optymalizowane przez sztuczne kolonie pszczół (algorytm ABC). Nunes de Castro, von Zuben [56] opisali system aiNet oparty o algorytmy sztucznych systemów immunologicznych do analizy danych. Prathyusha, Kannayaram [61] przedstawili nowatorski mechanizm oparty o AIS do mitygacji ataków sieciowych DDoS w chmurze. Szczególną pozycję w literaturze zajmują wzorce [37, 38, 77, 78, 79]. Wzorce pozwalają na optymalną generację zbioru receptorów na podstawie ciągów własnych. W związku z wyborem wzorców jako jednego z kluczowych zagadnień w pracy, zostaną one szczegółowo omówione w rodziale 3 w trakcie przedstawienia własnego rozwiązania.

Rozdział 3

System wykrywania intruzów

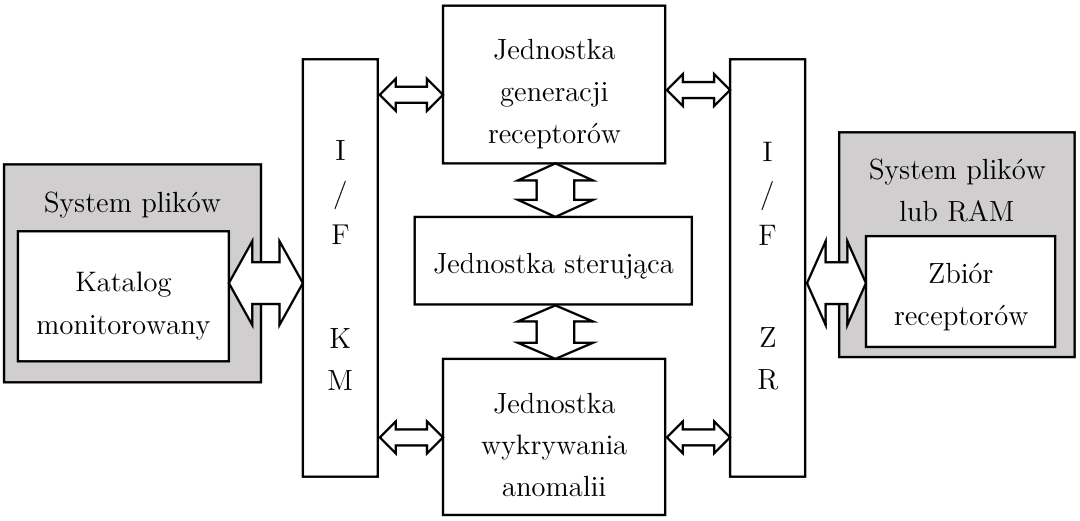
Proponuję własne rozwiązanie wykorzystujące zmodyfikowany algorytm negatywnej selekcji do wykrywania ingerencji w programy komputerowe w systemie operacyjnym. Rozwiązanie to stanowi system wykrywania intruzów (IDS) działający w systemie operacyjnym Windows.

Zaproponowany system monitoruje określoną lokalizację w systemie plików w celu wykrycia infekcji w programach komputerowych.

3.1 Schemat systemu i opis elementów

Schemat systemu znajduje się na rysunku 3.

W skład proponowanego systemu wchodzą: katalog monitorowany (KM) stanowiący zbiór programów do skanowania cyklicznego, zbiór receptorów (ZR) przechowujący receptory dla każdego monitorowanego programu, jednostka generacji receptorów (JGR) odpowiedzialna za generację zbiorów receptorów, jednostka wykrywania anomalii (JWA) wykonująca cykliczne skanowanie programów w katalogu monitorowanym, jednostka sterująca (JS) odpowiedzialna za wywoływanie JGR i JWA i bloki interfejsów (I/F) służące do komunikacji z systemem opera-



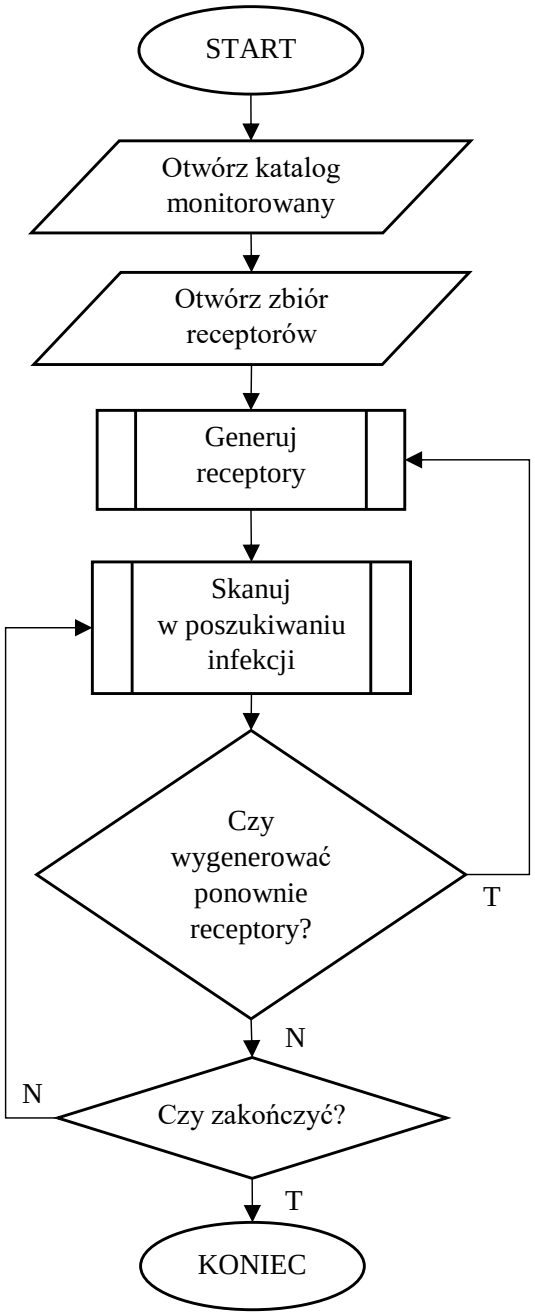
Rysunek 3. Schemat proponowanego systemu wykrywania intruzów

cyjnym w celu zapisu/odczytu KM i ZR.

Katalog monitorowany jest folderem w systemie plików zawierającym pliki programów, które system wykrywania intruzów ma cyklicznie skanować w poszukiwaniu infekcji. W systemie Windows mógłby to być na przykład folder C:\Windows, zawierający m.in. pliki wykonywalne niezbędne do działania systemu operacyjnego.

Zbiór receptorów jest miejscem przechowywania zestawu receptorów do każdego monitorowanego programu. Zbiór ten może być umieszczony w pamięci ulotnej lub nieulotnej, zależnie od preferencji administratora systemu. Na rysunku 4 znajduje się ogólny schemat blokowy działania IDS.

Jednostka sterująca jest blokiem systemu odpowiedzialnym za koordynację pracy pozostałych jednostek. Realizuje ona algorytm przedstawiony na rysunku 4. Po uruchomieniu rozwiązania JS otwiera przy pomocy interfejsów katalog monitorowany i próbuje uzyskać dostęp do zbioru receptorów. W przypadku niepowodzenia wykonania operacji otwarcia katalogu monitorowanego występuje błąd i program jest kończony natychmiastowo. Jeżeli zbiór receptorów jest pusty lub



Rysunek 4. Ogólny schemat blokowy działania IDS realizowany przez jednostkę sterującą

brakuje kompletu wygenerowanych wcześniej receptorów dla przynajmniej jednego programu w KM, następuje generacja receptorów. W przeciwnym wypadku algorytm przechodzi do cyklicznego skanowania każdego z programów monitorowanych. Jeżeli administrator systemu podejmie decyzję o regeneracji zbioru receptorów, system uruchamia procedurę pełnej regeneracji kompletu receptorów dla wszystkich programów z katalogu monitorowanego.

Jednostka generacji receptorów oferuje funkcjonalność utworzenia zbioru receptorów przyjmując, że programy w katalogu monitorowanym są niezainfekowane (są aktualnie w pożądanym stanie). Receptory generowane są dla określonych parametrów (długości receptora l oraz progu aktywacji m) zależnie od wybranej przez administratora metody: losowo lub z wykorzystaniem wzorców, przy zachowaniu reguły, że żaden z powstałych receptorów nie może wykrywać potem fragmentów oryginalnych jako zainfekowanych. W przypadku metody losowej, ciągi binarne receptorów powstają przez odpytanie generatora liczb pseudolosowych i konkatenację bajtów dopóki nie powstanie R_{max} ciągów o długości co najmniej l każdy. W przypadku metody wzorców, generacja polega na utworzeniu tablicy wzorców dla każdej kombinacji bitów progu aktywacji. Na podstawie tej tablicy i oryginalnych fragmentów programu generowane są następnie receptory.

Jednostka detekcji anomalii jest elementem systemu, który odpowiada za skanowanie programów monitorowanych w celu wykrycia infekcji. Skanowanie jednego programu polega na porównaniu każdego fragmentu programu z każdym receptorem z podzbioru receptorów odpowiadającemu danemu programowi. Do porównania owego wykorzystywany jest wzór wykorzystujący próg aktywacji m receptora. W przypadku dopasowania co najmniej m kolejnych bitów między sprawdzanym aktualnie fragmentem programu a receptorem mówimy, że doszło do aktywacji receptora i w konsekwencji, wykrycia infekcji.

3.2 Zasadność użycia NSA

Algorytm negatywnej selekcji wykorzystywany jest do wykrywania anomalii typowo w ruchu sieciowym między innymi ze względu na strumieniowy charakter danych do skanowania. W niniejszej pracy badane są jednak możliwości wykorzystania NSA do detekcji infekcji programów komputerowych przechowywanych lokalnie na komputerze niepodłączonym do sieci.

Systemy operacyjne, na przykład z rodzin Windows i Unix, zapisują i odczytują dane opierając się na systemach plików, które organizują dane w zbiory nazywane plikami. Pliki mają skończoną długość i przechowywane są fizycznie głównie na nośnikach pamięci nieulotnej, zwykle masowej. Programy komputerowe, które przeznaczone są do wykonania przez procesor umieszczone są zazwyczaj w formie listy instrukcji procesora w tzw. plikach wykonywalnych. Mimo, że fizyczne dane na nośniku pamięci nieulotnej mogą nie być umieszczone kolejno w momencie zapisu, to dzięki systemowi plików w trakcie odczytu z pliku wykonywalnego instrukcje wykonywane są przez procesor kolejno, lub jedna po drugiej. W przypadku odczytywania kolejno instrukcji programu komputerowego można mówić więc o istnieniu pewnego strumienia danych do skanowania, co z kolei sugeruje zasadność zastosowania algorytmu negatywnej selekcji do detekcji infekcji takiego programu.

Kolejną przesłanką sugerującą zasadność wykorzystania NSA jest jego zdolność do rozpoznawania nieznanych zagrożeń. Dzięki zdolności trenowania algorytmu NSA możliwe jest wykrycie infekcji, o których sygnaturach nie wiedział wcześniej IDS.

W trakcie prac nad rozprawą zmodyfikowane zostały metody NSA: losowa i wzorców. Testowanie wstępne metody losowej i wzorców pozwoliło wyciągnąć wniosek, że wzorce dają większe możliwości, dlatego więcej miejsca zostanie poświęcone w pracy na metodę wzorców.

3.3 Zaimplementowane metody

Aby system mógł wykrywać infekcje w katalogu monitorowanym, musi najpierw wygenerować zbiór receptorów. Zaproponowany system wykrywania intruzów udostępnia dwie metody generacji tego zbioru: losową i wzorców. W celu przeprowadzenia badań efektywności każdej z wybranych metod, w systemie istnieje możliwość włączenia lub wyłączenia autorskiej modyfikacji powodującej generację zbioru R_I, jednak zostanie ona opisana nieco dalej.

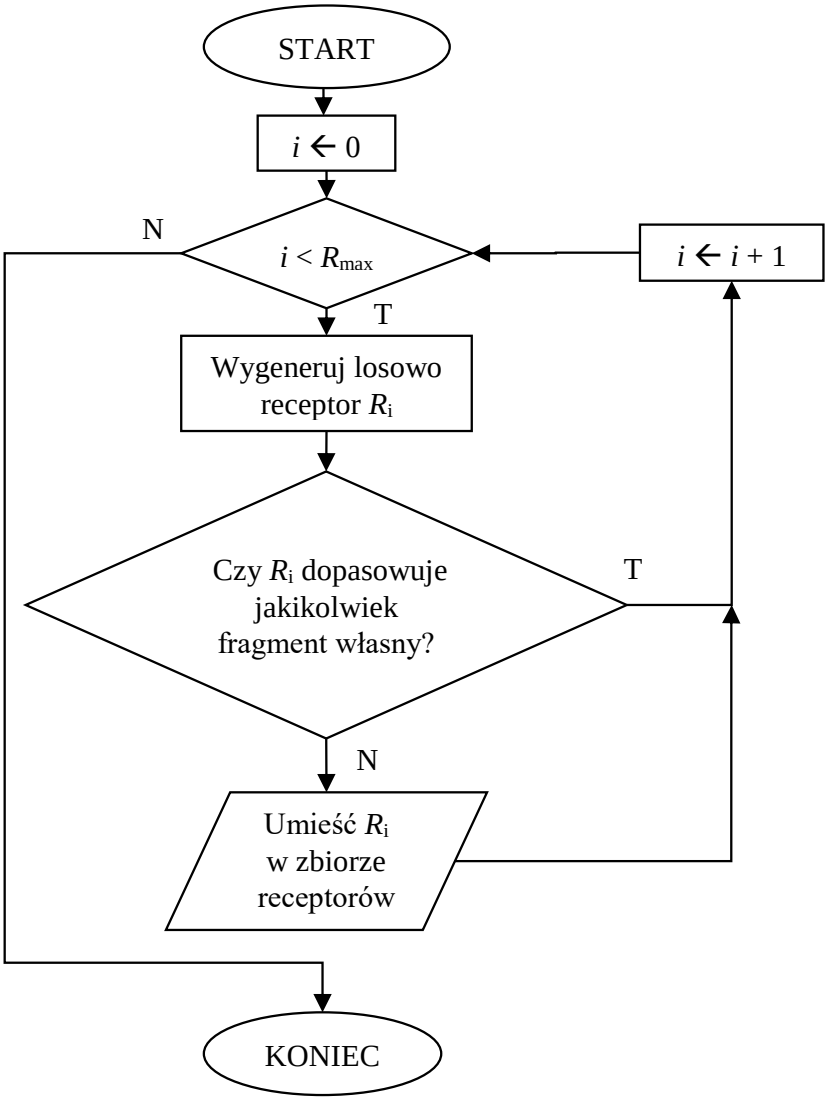
3.3.1 Metoda losowa

Podejście to zakłada utworzenie zbioru receptorów korzystając z generatora liczb pseudolosowych [26]. Na rysunku 5 przedstawiono schemat blokowy działania algorytmu losowej generacji receptorów w proponowanym IDS.

Parametrami wejściowymi do algorytmu losowej generacji receptorów są stałe: l (długość receptora w bitach), m (próg aktywacji receptora) oraz R_{max} (liczba receptorów do wygenerowania dla danego programu monitorowanego). Maksymalna liczba receptorów, jaką może wygenerować algorytm dla jednego programu określona jest równaniem:

$$max(R_{max}) = 2^l. \tag{1}$$

Jeden adres w pamięci RAM w architekturze x86 pozwala na dostęp do jednego bajta danych (8 bitów). W związku z tym, na etapie generacji właściwej do utworzenia każdego receptora w algorytmie skorzystano z generatora liczb pseudolosowych (RNG) do wytwarzania liczb ośmiobitowych bez znaku, czyli z zakresu $\langle 0; 255 \rangle$. Liczby te stanowią losowe, 8-bitowe części tworzonego receptora. Liczba części 8-bitowych, które należy wygenerować by utworzyć losowy receptor



Rysunek 5. Schemat blokowy algorytmu generacji losowej receptorów

l-bitowy oznaczona jest jako R_f i jest określona następującym wzorem:

$$R_f = ceil(div(l, 8)), \tag{2}$$

gdzie *ceil* to funkcja sufit, a $div(a, b)$ jest dzieleniem arytmetycznym liczby a przez liczbę b. Po wygenerowaniu razem R_f części, łączone są one na zasadzie

konkatenacji, co powoduje utworzenie jednego losowego receptora. Jeżeli długość l jest niepodzielna przez 8, to w receptorze takim figurują bity niepotrzebne. Bity te, choć są obecne fizycznie w komórkach pamięci RAM lub rejestrach procesora aktualnie zajmowanych przez receptor, nie są brane pod uwagę w dalszych operacjach algorytmicznych. Rysunki 6 i 7 ilustrują to zachowanie.

Jak widać na rysunku 6, wygenerowany losowo receptor dla parametru $l = 16$ zajmuje dokładnie 16 bitów miejsca w pamięci, czyli 2 bajty. Wygenerowane przez RNG zostały kolejno bajty: `DF` i `B7`. W wyniku konkatenacji utworzyły one następnie receptor 16-bitowy o wartości `DFB7`. Na rysunku 7 przedstawiono natomiast przypadek receptora utworzonego losowo dla parametru $l = 27$. Ponieważ jest to wartość niepodzielna przez 8, receptor zajmie w pamięci dodatkowe nieistotne 5 bitów, oznaczone na rysunku na czerwono i pogrubioną czcionką. Wylosowane przez RNG bajty `AF`, `94`, `7B` i `6F` zostały skonkatenowane i utworzony został kandydat na receptor `AF947B6F`, który ze względu na 5 nieistotnych bitów de facto ma postać `AF947B60`, jeśli przyjmiemy, że bity nieistotne zawsze mają wartość 0.

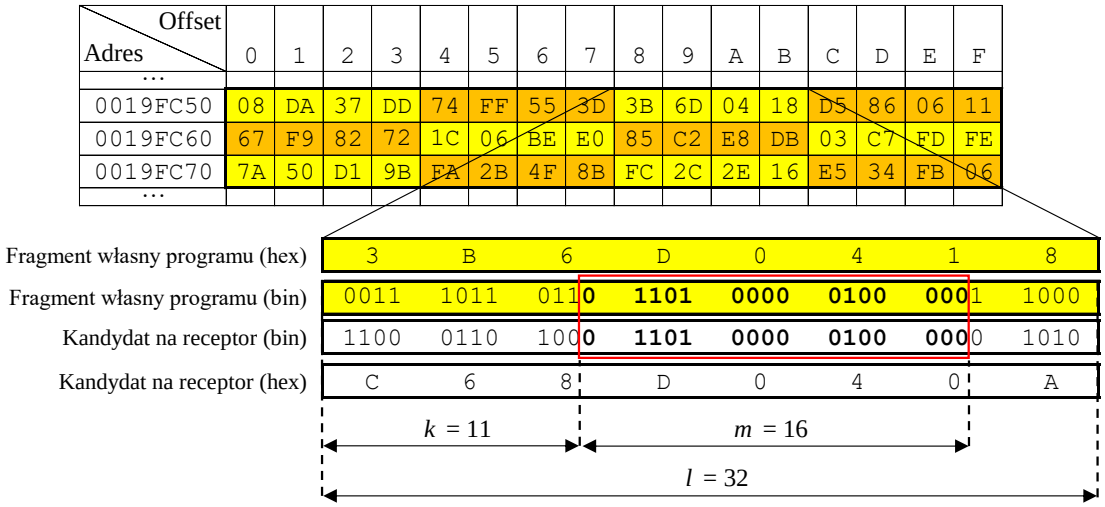
Każdy nowo wygenerowany ciąg stanowiący kandydata na receptor porównywany jest z każdym fragmentem własnym monitorowanego programu. Jeżeli kandydat zostanie dopasowany do przynajmniej jednego fragmentu własnego, to nie może zostać receptorem i nie jest umieszczany w zbiorze receptorów danego pro-

Receptor (hex)	DFB7	wygenerowany dla l = 16	Rf = 2
Organizacja receptora w RAM lub w rejestrze CPU			
Cyfra (bin)	1101 1111	1011 0111	
Cyfra (hex)	D F	B 7	
	bajt 1, MSB	*bajt 0, LSB*	

Rysunek 6. Przykład receptora wygenerowanego dla l podzielnego przez 8

Receptor (hex)	AF947B6F		wygenerowany dla l = 27		Rf = 4		
Organizacja receptora w RAM lub w rejestrze CPU							
Cyfra (bin)	1010	1111	1001	0100	0111	1011	0110 1111
Cyfra (hex)	A	F	9	4	7	B	6 F
	bajt 3, MSB		*bajt 2*		*bajt 1*		*bajt 0, LSB*

Rysunek 7. Przykład receptora wygenerowanego dla l niepodzielnego przez 8



Rysunek 8. Dopasowanie fragmentu własnego przez kandydata na receptor – zdarzenie eliminujące kandydata z puli potencjalnych receptorów

gramu monitorowanego. Na rysunku 8 przedstawiono przykładowe dopasowanie fragmentu własnego programu `3B6D0418` przez kandydata na receptor `C68D040A` dla wartości $l = 32$, $m = 16$ i adresu programu `0019FC58`. Jak można zauważyć, występuje tutaj dopasowanie wszystkich $m = 16$ bitów progu aktywacji (przy przesunięciu okna $k = 11$), co skutkuje aktywacją receptora dla fragmentu własnego programu czyli wykryciem kodu własnego jako infekcji. Ze względu na to kandydat na receptor `C68D040A` zostaje odrzucony i następuje generacja innego ciągu binarnego.

W momencie osiągnięcia przez zmienną iteracyjną i wartości R_{max} algorytm kończy generację receptorów dla danego programu monitorowanego.

3.3.2 Metoda wzorców

Jedną z metod, które powstały w celu selekcji receptorów lepszej niż losowo jest metoda wzorców zaproponowana między innymi w [78, 79]. Zaproponowany system wykrywania intruzów zakłada jej zastosowanie w celu optymalnej generacji receptorów.

Wzorcem nazwiemy ciąg binarny, który posiada w sobie bity istotne i bity nieistotne. Bity istotne mogą mieć wartość 0 lub 1 i umieszczone są we wzorcach kolejno, a bity nieistotne oznaczane są we wzorcach gwiazdką (∗). Ciąg bitów istotnych będziemy nazywać sekwencją stałą wzorca.

Podstawą tej metody jest utworzenie specjalnej tablicy nazywaną tablicą wzorców, oznaczoną jako **T**. W tablicy tej umieszczone są wzorce o określonej strukturze. Struktura wzorców, zgodnie z definicją, zależy od liczby bitów istotnych i liczby bitów nieistotnych. W tablicy **T** występują wzorce o długości l bitów (czyli takiej samej, jak receptory do wygenerowania). Liczba bitów istotnych we wzorcu jest równa progowi aktywacji m. Wynika z tego więc, że we wzorcu istnieje $l-m$ bitów nieistotnych. Liczba wierszy w tablicy **T** jest równa 2^m, a liczba kolumn ze wzorcami wynosi $l - m + 1$. Liczbę wzorców w **T** oznaczamy jako C_T i wynosi więc ona:

$$C_T = (l - m + 1) \cdot 2^m. \tag{3}$$

Dla typowych dla tej pracy parametrów takich jak $l = 32$, $m = 8$ oraz $l = 16$, $m = 8$ liczba C_T wynosi odpowiednio 6400 i 2304. W związku z tym, w celu przedstawienia sposobu powstawania tablicy **T** i omówienia kolejnych kroków w przystępny sposób, przyjęto, że na potrzeby omówienia metody wzorców wykorzystane będą parametry $l = 6$ i $m = 4$, co daje $C_T = 48$. Tablica **T** powstała dla tych parametrów znajduje się w Tabeli 1.

Pierwsza kolumna tablicy stanowi numer wiersza i. Każdy wiersz tablicy

Tabela 1. Tablica wzorców **T** powstała dla $l = 6$ i $m = 4$

Numer wiersza *i*	**Sekwencja stała o długości *m***	**T[*i*, 1]**	**T[*i*, 2]**	**T[*i*, 3]**
1	0000	0000**	*0000*	**0000
2	0001	0001**	*0001*	**0001
3	0010	0010**	*0010*	**0010
4	0011	0011**	*0011*	**0011
5	0100	0100**	*0100*	**0100
6	0101	0101**	*0101*	**0101
7	0110	0110**	*0110*	**0110
8	0111	0111**	*0111*	**0111
9	1000	1000**	*1000*	**1000
10	1001	1001**	*1001*	**1001
11	1010	1010**	*1010*	**1010
12	1011	1011**	*1011*	**1011
13	1100	1100**	*1100*	**1100
14	1101	1101**	*1101*	**1101
15	1110	1110**	*1110*	**1110
16	1111	1111**	*1111*	**1111

zawiera w drugiej kolumnie sekwencję stałą odpowiadającą binarnej formie liczby $(i - 1)$, zapisanej na m bitach. Kolumny **T**[i, j], gdzie j jest przesunięciem sekwencji stałej plus 1, zawierają l-bitowe wzorce powstałe poprzez umieszczenie sekwencji stałej na pozycji przesuniętej w prawo o $(j - 1)$ bitów i wypełnienie pozostałych wartości bitami nieistotnymi, czyli gwiazdkami. W rezultacie na podstawie tabeli 1 wzorzec **T**[1, 1] ma postać `0000**`, a na przykład wzorzec **T**[7, 3] = `**0110`.

Analogicznie jak w przypadku metody losowej, tak i tutaj należy upewnić się, że żaden potencjalny receptor nie spowoduje wykrycia infekcji we fragmentach

własnych. W proponowanym rozwiązaniu fragmenty własne odczytywane są bezpośrednio z pliku programu monitorowanego. Na potrzeby kontynuacji ilustracji działania metody wzorców zdefiniowany zostanie zbiór **S**, zawierający fragmenty własne których niewykrywalność przez kandydatów na receptory będzie stanowić warunek ich akceptacji przez algorytm. Niech zbiór **S** ma więc przykładową postać:

$$\begin{aligned}\mathbf{S} = \{&101011, 101100, 001101, 111011, 000001, 111000,\\ &010110, 011100, 000111, 010001, 110001\}.\end{aligned} \tag{4}$$

Niezbędne jest teraz zidentyfikowanie wzorców, których nie można wykorzystać do skonstruowania receptorów. W tym celu tworzona jest tablica filtrująca $\mathbf{T_f}$. Tabela 2 przedstawia tę tablicę.

Zasada generacji tablicy $\mathbf{T_f}$ jest następująca. Każdy wzorzec z tablicy **T** porównywany jest ze wszystkimi fragmentami własnymi ze zbioru **S**. Jeżeli wzorzec **T**[i, j] zostanie dopasowany z którymkolwiek z fragmentów własnych ze zbioru **S** (dopasowanie polega na obecności sekwencji stałej ze wzorca na tej samej pozycji we fragmencie własnym), to taki wzorzec się nie nadaje na receptor i do tablicy $\mathbf{T_f}$[i, j] wpisywana jest wartość 0. Jeżeli wzorzec nie dopasuje żadnego fragmentu własnego to $\mathbf{T_f}$[i, j] = 1. Na przykład wzorzec **T**[4, 1] = `0011**` dopasowuje fragment własny $\mathbf{S}_3$ = `001101`, więc $\mathbf{T_f}$[4, 1] = 0, natomiast ponieważ wzorzec **T**[11, 2] = `*1010*` nie dopasowuje żadnego fragmentu własnego, to $\mathbf{T_f}$[11, 2] = 1.

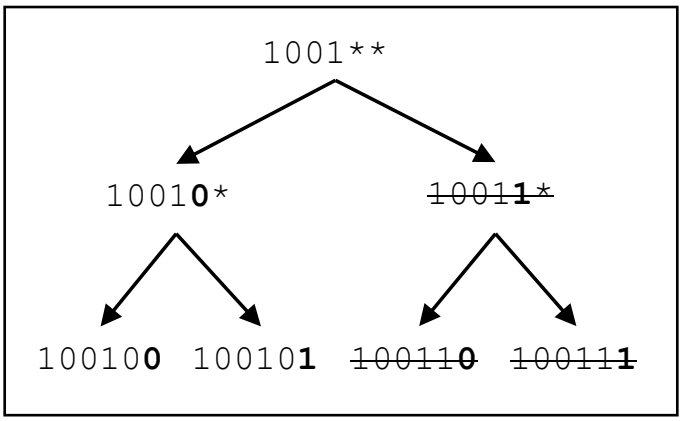
Wiedząc, że odfiltrowane zostały już wszystkie wzorce nienadające się na receptory, na podstawie tablic **T** i $\mathbf{T_f}$ można na tym etapie rozpocząć budowanie zbioru receptorów. Odbywa się to przy pomocy reprezentacji wzorców jako drzew binarnych. Ponieważ każdy zakwalifikowany wzorzec ma w tym przykładzie po 2 bity nieistotne, z pojedynczego wzorca mogą powstać maksymalnie 4 receptory. Na rysunku 9 przedstawiono przykład generacji receptorów z jednego wzorca:

Tabela 2. Tablica filtrująca $\mathbf{T_f}$ utworzona podstawie tablicy **T** i zbioru **S**

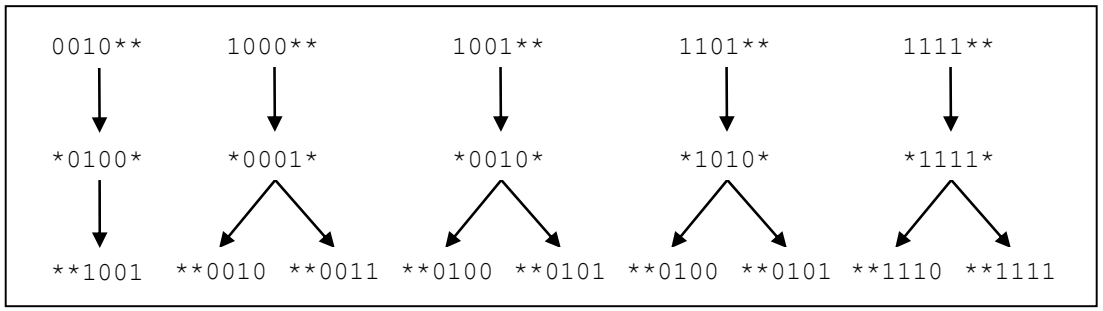
Numer wiersza i	**Sekwencja stała o długości** m	**T[**i**, 1]**	**T[**i**, 2]**	**T[**i**, 3]**
1	0000	0	0	1
2	0001	0	1	0
3	0010	1	1	1
4	0011	0	0	1
5	0100	0	1	1
6	0101	0	0	1
7	0110	1	0	0
8	0111	0	1	0
9	1000	1	0	0
10	1001	1	1	1
11	1010	0	1	1
12	1011	0	0	0
13	1100	0	0	0
14	1101	1	0	0
15	1110	0	0	1
16	1111	1	1	1

`1001**`.

Jak wynika z rysunku, korzeniem drzewa binarnego jest wzorzec `1001**`. Z korzenia tworzone są dwie gałęzie. W pierwszej gałęzi pierwszy nieistotny bit zamieniono na 0 uzyskując `10010*`, a w drugiej gałęzi – na jedynkę (`10011*`). Analogicznie, od każdej z utworzonych gałęzi tworzy się poddrzewa do momentu zamiany wszystkich bitów nieistotnych na zera i jedynki. Jak łatwo zauważyć, najniższy poziom całego drzewa ma łącznie 4 liście, które stanowią potencjalne receptory. Nie wszystkie kombinacje te mogą jednak faktycznie stać się receptorami - gałęzie i liście niedozwolone przekreślono na rysunku. Gałąź `10011*`



Rysunek 9. Drzewo binarne obrazujące wszystkie możliwe kombinacje i wszystkie dozwolone kombinacje receptorów ze wzorca `1001**`



Rysunek 10. Drzewa binarne generacji receptorów w przykładzie

nie może utworzyć żadnych receptorów, ponieważ dopasowuje wzorzec $\mathbf{T}[4, 2]$ = `*0011*`, który został oznaczony jako niedozwolony w tabeli filtrującej: $\mathbf{T_f}[4, 2] = 0$. Z tego względu pretendenci na receptory `100110` i `100111` zostają odrzuceni. Jednocześnie gałąź `10010*`, która dopasowuje wzorzec $\mathbf{T}[3, 2]$ = `*0010*` (dozwolony dzięki $\mathbf{T_f}[3, 2] = 1$), tworzy liście `100100` i `100101`, które dopasowują odpowiednio wzorce dozwolone $\mathbf{T}[5, 3]$ i $\mathbf{T}[6, 3]$. Liście te są na tym etapie kompetentnymi receptorami i mogą być umieszczone w ostatecznym zbiorze receptorów.

Na rysunku 10 przedstawiono wszystkie drzewa binarne dla analizowanego przykładu ze zbiorem $\mathbf{S}$ i tablicami $\mathbf{T}$ i $\mathbf{T_f}$. Drzewa te zawierają wyłącznie do-

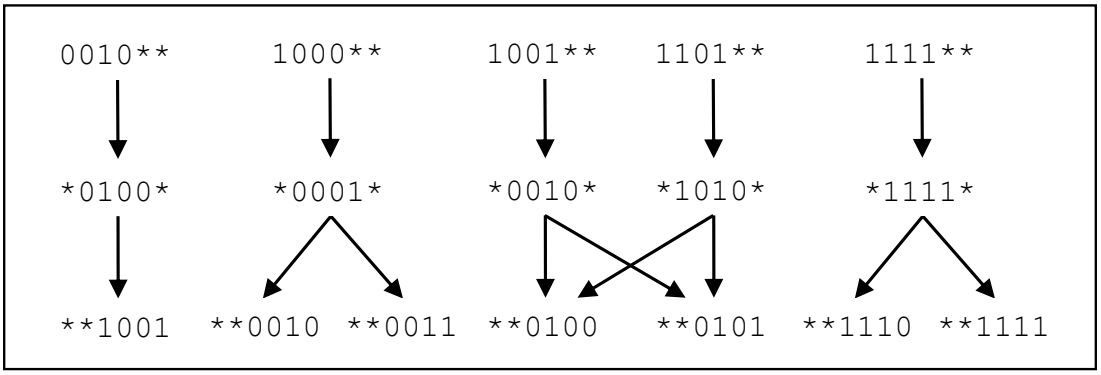
zwolone wzorce na podstawie tablicy $\mathbf{T_f}$. Jak widać na rysunku, możliwa jest generacja 9 receptorów. Każdą gwiazdkę należy zastąpić odpowiadającym jej bitem na poprzednim poziomie. Wygenerowany w ten sposób zbiór receptorów ma więc postać:

$$\begin{aligned}\mathbf{R} = \{&001001, 100010, 100011, 100100, 100101,\\ &110100, 110101, 111110, 111111\}.\end{aligned} \tag{5}$$

Liczba wygenerowanych receptorów R_n wynosi więc 9. Przyjrzyjmy się rysunkowi 11 i wygenerowanemu zbiorowi receptorów $\mathbf{R}$. Można zauważyć, że gałęzie `10010*` i `11010*` mają takie same liście `**0100` i `**0101`. Jest to wspólne dla tych dwóch gałęzi poddrzewo, które indukuje receptory: `100100`, `100101`, `110100`, `110101`. Załóżmy ciąg binarny $d \notin \mathbf{S}$, który pasuje do wzorca `**0100`. Ciąg d jest wykrywany zarówno przez receptor `100100`, jak i przez receptor `110100`, co wskazuje na istnienie w zbiorze $\mathbf{R}$ receptorów zbędnych (nadmiarowych).

W celu optymalizacji zbioru $\mathbf{R}$, w drzewach binarnych poszukiwane są wspólne poddrzewa [78], a następnie łączone tak, by wśród liści (tj. na najniższym poziomie) nie znajdowały się duplikaty wzorców. Ponieważ w naszym przykładzie na rysunku 10 duplikatami były liście `**0100` i `**0101`, należało scalić gałęzie `10010*` i `11010*`. Na rysunku 11 przedstawiono drzewa binarne po scaleniu wspólnych poddrzew.

Zbiór $\mathbf{R}$ jest optymalny, jeżeli do konstrukcji receptorów zostaną wykorzystane wszystkie liście, z zachowaniem zasady, że dany liść można wykorzystać tylko raz [77]. Jednocześnie każdy korzeń drzew binarnych powinien zaczynać przynajmniej jedną ścieżkę do liścia. Jak widać na rysunku 11, optymalna liczba receptorów nie powinna przekraczać więc $R_n = 7$. Ostateczna postać zbioru $\mathbf{R}$ zależy od obranej strategii wyboru ścieżki w drzewie (na przykład, do wzorca



Rysunek 11. Drzewa binarne po scaleniu wspólnych poddrzew

0100 można dojść na dwa sposoby, tak samo dla wzorca **0101). Możliwymi postaciami zbioru **R są więc:

- $\mathbf{R}_A$ = { 001001, 100010, 100011, 100100, 110101, 111110, 111111 },
- $\mathbf{R}_B$ = { 001001, 100010, 100011, 100100, 100101, 111110, 111111 },
- $\mathbf{R}_C$ = { 001001, 100010, 100011, 110100, 110101, 111110, 111111 },
- $\mathbf{R}_D$ = { 001001, 100010, 100011, 110100, 100101, 111110, 111111 }.

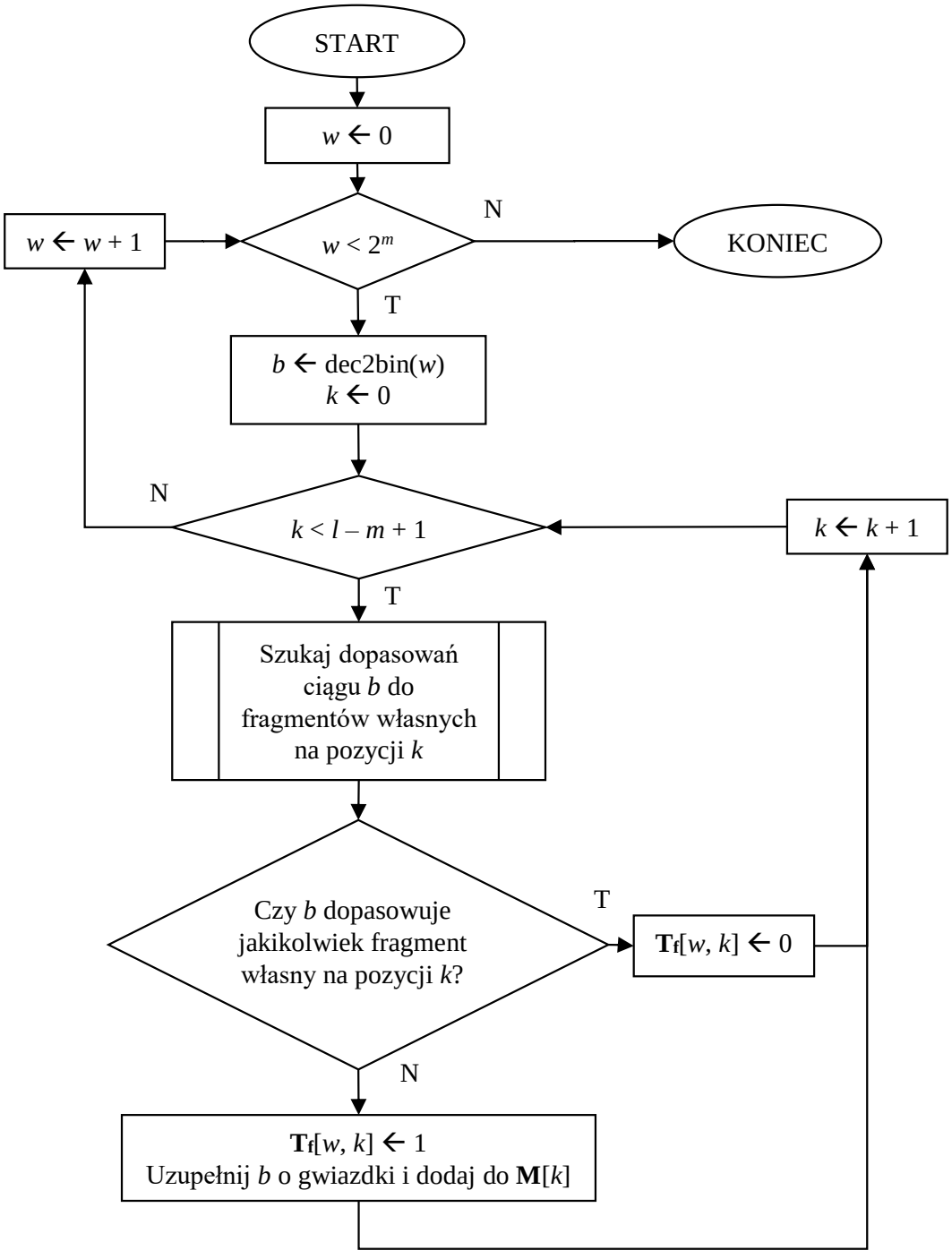
Na rysunku 13 przedstawiono schemat blokowy generacji tablicy filtrującej $\mathbf{T_f}$. Numer wiersza oznaczono przez w, sekwencję stałą przez b, a przesunięcie okna przez k. $\mathbf{M}[k, n]$ jest tablicą pomocniczą, zawierającą wszystkie wzorce dla poziomu k, które są dozwolone przez $\mathbf{T_f}[w, k]$, a n oznacza liczbę porządkową wzorca do którego się odwołujemy. Funkcja $dec2bin(w)$ odpowiada za konwersję wartości numeru wiersza z systemu dziesiętnego do ciągu binarnego. Na rysunku 12 przedstawiono przykładowe dopasowanie między ciągiem sprawdzanym a receptorem dla $l = 8$, $m = 4$.

Na rysunku 14 przedstawiono schemat blokowy generacji drzew binarnych wzorców redukującej także nadmiarowe receptory. Numer wiersza w **T** i $\mathbf{T_f}$

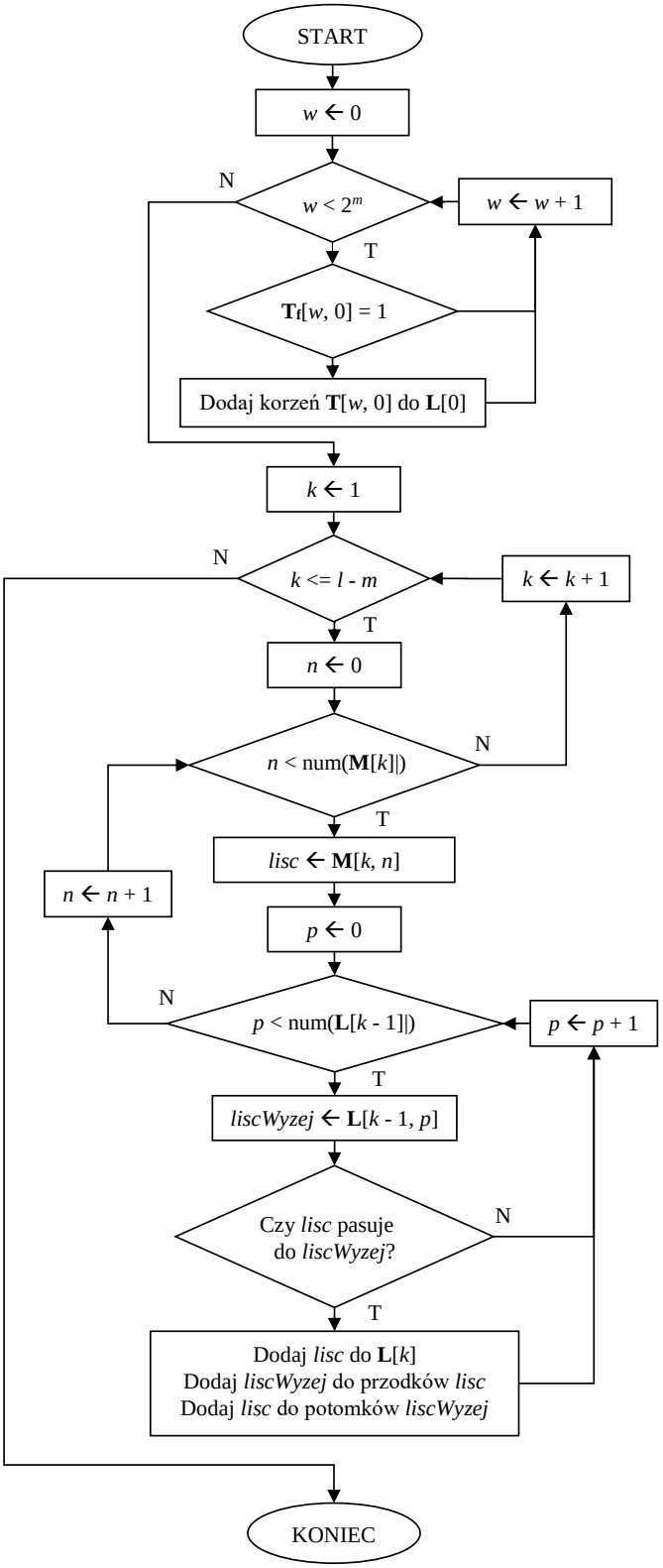
Sprawdzany ciąg	100**0101**0
Receptor	011**0101**1

Rysunek 12. Przykładowe dopasowanie między ciągiem sprawdzanym a receptorem dla $l = 8$, $m = 4$, $k = 3$

oznaczono przez w, wyjściową tablicę drzew binarnych przez $\mathbf{L}[i, j]$ gdzie przez i oznaczono poziom drzewa (0 = poziom korzeni) a przez j oznaczono gałąź/liść do którego się odwołujemy na danym poziomie, przesunięcie okna przez k, długość receptorów przez l. $\mathbf{M}[k, n]$ opisano wyżej. Zmienna *lisc* przechowuje n-ty wzorzec dozwolony na poziomie k, a zmienna *liscWyzej* przechowuje p-ty wzorzec dozwolony na poziomie wyższym, czyli $k - 1$. Liście posiadają wiedzę na temat swoich przodków i potomków. Funkcja $num(\mathbf{Z})$ oznacza moc danego zbioru $\mathbf{Z}$. Redukcja liczby receptorów skutkuje zmniejszeniem zajętości pamięci.



Rysunek 13. Schemat blokowy generacji tablicy filtrującej $\mathbf{T_f}$



Rysunek 14. Schemat blokowy generacji drzew binarnych wzorców

3.4 Motywacja do wprowadzenia modyfikacji do algorytmów

Instrukcje programów komputerowych zawarte w plikach zapisane są w kodzie maszynowym. Na rysunku 15 przedstawiono przykładowe instrukcje typowego programu wykonywalnego w systemie operacyjnym Windows. Adresy indywidualnych bajtów tych instrukcji w pliku rozbito dla czytelności na część bazową adresu i przesunięcie (offset) i aby uzyskać adres indywidualny należy te części zsumować – przykładowo, pod adresem `0019FC50 + 0` (czyli `0019FC50`) znajduje się bajt `08`, a pod adresem `0019FC60 + D` (czyli `0019FC6D`) znajduje się bajt `C7`.

Offset / Adres	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
...																
0019FC50	08	DA	37	DD	74	FF	55	3D	3B	6D	04	18	D5	86	06	11
0019FC60	67	F9	82	72	1C	06	BE	E0	85	C2	E8	DB	03	C7	FD	FE
0019FC70	7A	50	D1	9B	FA	2B	4F	8B	FC	2C	2E	16	E5	34	FB	06
...																

Rysunek 15. Instrukcje przykładowego programu zapisane w kodzie maszynowym (wartości podane w systemie szesnastkowym)

Tradycyjnie, algorytm NSA wykorzystuje jeden zbiór receptorów [26]. W przypadku wykrywania anomalii w ruchu sieciowym może się to okazać wystarczające, ponieważ anomalie nie mogą wystąpić między pakietami sieciowymi. Infekcje w lokalnych programach komputerowych mogą mieć natomiast lokalizacje, które dla wystarczająco małej liczby zmodyfikowanych przez intruza kolejnych bitów powodują niewidoczność dla algorytmów detekcji anomalii opartych na NSA.

Rozważmy przypadek anomalii o małym rozmiarze, występującej między fragmentami programu o przykładowej długości $l = 4$ B (32 bity). Przypadek ten zilustrowano na rysunku 16. W celu poprawienia czytelności rysunku, 32-bitowe fragmenty własne programu oznaczono naprzemiennie na pomarańczowo i żółto.

Adres \ Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
...																
0019FC50	08	DA	37	DD	74	FF	55	3D	3B	6D	04	18	D5	86	06	11
0019FC60	67	F9	82	72	1C	06	BE	E0	85	C2	E8	DB	03	C7	FD	FE
0019FC70	7A	50	D1	9B	FA	2B	4F	8B	FC	2C	2E	16	E5	34	FB	06
...																

Adres \ Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
...																
0019FC50	08	DA	37	DD	74	FF	55	3D	3B	6D	04	18	D5	86	06	11
0019FC60	67	F9	82	**7A**	**8C**	06	BE	E0	85	C2	E8	DB	03	C7	FD	FE
0019FC70	7A	50	D1	9B	FA	2B	4F	8B	FC	2C	2E	16	E5	34	FB	06
...																

Rysunek 16. Przykładowy program podzielony częściowo na 32-bitowe fragmenty: a) przed pojawieniem się anomalii, b) po pojawieniu się anomalii pod adresami `0019FC63`, `0019FC64`

Kolory te pokazują, które bajty należą do danego 32-bitowego fragmentu programu. Na czerwono i czcionką pogrubioną oznaczono anomalię wprowadzoną do programu przez intruza. Ponieważ odczytywane przez algorytm fragmenty własne programu zawsze mają taką samą długość co receptory (w tym przypadku $l = 32$), może się okazać, że ta konkretna anomalia (bajt na czerwono o wartości efektywnej `0A80`) nie zostanie wykryta przez żaden receptor 32-bitowy ani pod adresem `0019FC60` (żaden receptor nie „zauważy” cyfry zainfekowanej `A`), ani pod adresem `0019FC64` (żaden receptor nie „zauważy” cyfry zainfekowanej `8`).

3.5 Opis modyfikacji algorytmów – dodatkowy zbiór receptorów

Powyższy przypadek był motywacją do znalezienia rozwiązania tego problemu. W konsekwencji w algorytmie negatywnej selekcji wprowadzono autorską modyfikację w celu poprawienia wykrywalności infekcji. Modyfikacja polega na wprowadzeniu do algorytmu dodatkowego zbioru receptorów, nazywanego zbiorem receptorów międzykomórkowych (lub zbiorem receptorów pomocniczych) i oznaczanego R_I. Zbiór ten generowany jest na etapie tworzenia receptorów na podstawie fragmentów programu, które znajdują się między komórkami fragmentów głównych. W celu omówienia modyfikacji wykorzystane będą parametry $l = 32$, $m = 16$.

Adres \ Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
…																
0019FC50	08	DA	37	DD	74	FF	55	3D	3B	6D	04	18	D5	86	06	11
0019FC60	67	F9	82	7A	8C	06	BE	E0	85	C2	E8	DB	03	C7	FD	FE
0019FC70	7A	50	D1	9B	FA	2B	4F	8B	FC	2C	2E	16	E5	34	FB	06
…																

Rysunek 17. 32-bitowe fragmenty własne programu (na przemian na pomarańczowo i na żółto) i 16-bitowe fragmenty międzykomórkowe programu (pogrubione obramowanie)

Na rysunku 17 przedstawiono w miejscach pogrubionych lokalizacje 16-bitowych fragmentów programu, na podstawie których możliwa była generacja zbioru R_I. Zbiór ten posiada własne parametry analogiczne do l i m. Liczba bitów receptorów międzykomórkowych oznaczana jest jako l_I, a próg aktywacji przez m_I. Jak łatwo zauważyć, w tym przypadku każdy 16-bitowy fragment składa się z najmłodszego bajtu oryginalnego fragmentu 32-bitowego (który to bajt stał się teraz najstarszym) i z najstarszego bajtu kolejnego fragmentu 32-bitowego (który to bajt stał się teraz najmłodszym). Przykładowo, pod adresem `0019FC67` powstał

16-bitowy fragment międzykomórkowy o wartości `E085`, a pod adresem `0019FC6F` powstał fragment o wartości `FE7A`. Starszy bajt 16-bitowego fragmentu spod adresu `0019FC4F` i młodszy bajt fragmentu spod adresu `0019FC7F` nie zostały pokazane na rysunku, ale nie są istotne.

Uogólniając, zbiór R_I który utworzono na etapie generacji receptorów, wykorzystywany jest na etapie wykrywania anomalii do sprawdzania, czy nie wystąpiły anomalie pomiędzy komórkami l-bitowymi. Ważne jest spostrzeżenie, że zbiór R_I nadaje się tylko i wyłącznie do weryfikacji (skanowania w poszukiwaniu infekcji) l_I-bitowych fragmentów międzykomórkowych, analogicznie jak w przypadku tradycyjnego zbioru receptorów (który nadaje się tylko do skanowania l-bitowych kolejnych fragmentów programu). Uwzględnienie zbioru R_I w procesie detekcji zwiększa prawdopodobieństwo wykrycia infekcji, ale wydłuża także na etapie tworzenia receptorów czas generacji.

3.6 Przewidywany wpływ modyfikacji na osiągi IDS

Przewidywane jest, że wprowadzenie zbioru receptorów pomocniczych wpłynie pozytywnie na prawdopodobieństwo wykrywalności anomalii przez IDS. Wykazanie, że modyfikacja ma co najmniej korzystny wpływ na wykrywalność infekcji, a na pewno nie szkodliwy, jest łatwe.

Zakładając, że A jest zdarzeniem że anomalia została znaleziona dzięki receptorom podstawowym, a B jest zdarzeniem że anomalia została znaleziona dzięki receptorom pomocniczym, $P(A)$ jest prawdopodobieństwem znalezienia anomalii przez receptory podstawowe, a $P(B)$ przez receptory pomocnicze. $P(A \cup B)$ jest prawdopodobieństwem, że anomalia została znaleziona dzięki receptorom podstawowym lub pomocniczym. $P(A \cap B)$ jest prawdopodobieństwem, że anomalia została znaleziona jednocześnie dzięki receptorom podstawowym i pomocniczym.

Wykluczając część wspólną $P(A \cap B)$ dzięki obecności zbioru receptorów międzykomórkowych (zdarzenie B) ogólne prawdopodobieństwo wykrycia anomalii ($P(A \cup B)$) może być albo równe $P(A)$ albo wyższe. Pozwala to stwierdzić, że modyfikacja na pewno nie zaszkodzi wykrywalności anomalii, co pozwoli na osiągnięcie wyników wykrywalności porównywalnych lub lepszych względem wyników znanych z literatury.

Mówiąc o prawdopodobieństwach wykrycia anomalii, nie sposób nie przytoczyć także kilku podstawowych wzorów, które mogą być wykorzystywane do analitycznego obliczania niektórych parametrów. Jeżeli receptory generowane są losowo, to ogólne prawdopodobieństwo dopasowania losowego ciągu przez receptor jest równe:

$$p(l, m, a) = \frac{1}{a^m} \cdot \left(\frac{(l - m)(a - 1)}{a} + 1 \right), \tag{6}$$

gdzie l jest długością receptora w bitach, m jest wartością progu aktywacji recep-

tora w bitach, a a jest mocą zbioru alfabetu używanego do konstrukcji receptora. Ważnym szczegółem jest, że równanie 6 jest prawdziwe tylko jeśli $m \geqslant \frac{l}{2}$. Ponieważ system IDS operuje na ciągach binarnych, alfabet ciągów zawiera tylko wartości 0 i 1. Oznacza to, że moc zbioru alfabetu wynosi 2, co pozwala na uproszczenie wzoru (6):

$$p(l,m) = \frac{1}{2^m} \cdot \left(\frac{l-m}{2} + 1 \right). \tag{7}$$

Na podstawie (7) można obliczyć ile ciągów binarnych jest w stanie być wykrytych przez pojedynczy receptor. Wartość ta jest stała dla wszystkich receptorów dla określonych wartości l i m. Liczba ciągów binarnych ze zbioru 2^l, rozpoznawanych przez pojedynczy receptor jest równa:

$$D(l,m) = p(l,m) \cdot 2^l, \tag{8}$$

co może być rozszerzone do:

$$D(l,m) = 2^{l-m-1} \cdot (2 + l - m). \tag{9}$$

Dla danego p, możemy określić prawdopodobieństwo niewykrycia anomalii przez żaden receptor. Aby to zdarzenie miało miejsce, musiałby wystąpić brak dopasowania okna progu aktywacji dla każdego receptora w ostatecznym zbiorze $\mathbf{R}$. W takim razie, prawdopodobieństwo, że żaden receptor nie dopasuje anomalii jest następujące:

$$p_f(l,m) = (1 - p(l,m))^{|R|}, \tag{10}$$

gdzie $|R|$ oznacza moc wygenerowanego zbioru receptorów. Prawdopodobieństwo sukcesu (zdarzenie, że co najmniej jeden receptor ze zbioru dopasuje anomalię)

jest różnicą p_f i prawdopodobieństwa zdarzenia prawie pewnego:

$$p_s(l, m) = 1 - p_f(l, m). \tag{11}$$

Należy podkreślić, że równania (10) i (11) mają zastosowanie tylko dla zbiorów receptorów które zostały wygenerowane losowo. Zastosowanie tych równań w [76] pozwoliło na proste analityczne obliczenie spodziewanych wartości prawdopodobieństw i porównanie ich z wartościami osiągniętymi w wyniku badań. W niniejszej pracy jednak uwaga zwrócona będzie głównie ku badaniom eksperymentalnym, które dzięki szerokiemu wachlarzowi parametrów wejściowych będą w stanie zademostrować skuteczność proponowanego IDS.

Obliczenie prawdopodobieństw wykrycia anomalii jest stosunkowo proste dla losowej generacji receptorów, ale w przypadku generacji na podstawie wzorców nie jest adekwatne. W przypadku metody wzorców analityczne oszacowanie prawdopodobieństw wykrycia anomalii jest trudne [77], więc w literaturze oblicza się liczbę dziur [37] powstałych po procesie generacji receptorów, i na jej podstawie próbuje oszacować przewidywaną wykrywalność. Szacunków tych dokonuje się [77] przy pomocy wzoru:

$$p_f = \frac{|S|+n}{2^l}, \tag{12}$$

gdzie $|S|$ to moc zbioru elementów własnych, l to długość receptora w bitach, a n to liczba dziur. Jednak, ostatecznie w celu porównania dostępnych w literaturze metod robi się zwykle ekstensywne badania eksperymentalne.

Ze względu na konieczność przechowania większej liczby receptorów niż w przypadku metod niezmodyfikowanych, przewidywane jest, że obecność receptorów międzykomórkowych wpłynie na wzrost zajętości pamięci. Wzrost zajętości pamięci powinien być proporcjonalny do wzrostu liczby receptorów międzykomórkowych.

Rozdział 4

Badania eksperymentalne

Proponowany w niniejszej pracy system wykrywania intruzów został zaimplementowany i dogłębnie przetestowany dla szerokiego wachlarza parametrów wejściowych.

4.1 Testowanie IDS

4.1.1 Konfiguracja

W celu przeprowadzenia badań, system wykrywania intruzów został zaimplementowany w języku C++ i skompilowany przy pomocy kompilatora MSVC dołączanego do zintegrowanego środowiska programistycznego Microsoft Visual Studio 2019. Wszystkie testy przeprowadzono na stacji roboczej o konfiguracji umieszczonej w Tabeli 3.

Na partycji C: zawierającej system operacyjny utworzono folder o nazwie ids, który ustawiono w systemie wykrywania intruzów na lokalizację do monitorowania (C:\ids). W związku z faktem, że IDS monitoruje wszystkie pliki w zadanym folderze, niezbędne było uzyskanie pliku do monitorowania. W tym celu napisano w języku C++ program wyświetlający na ekranie wiadomość MessageBox "This

Tabela 3. Konfiguracja stacji roboczej do badań eksperymentalnych

Element	**Wartość**
System operacyjny (OS)	Microsoft Windows 10 LTSC
Architektura OS	64-bit
Procesor (CPU)	Intel Core i9-10900K
Taktowanie CPU	3.7 GHz
Liczba rdzeni CPU	10
Liczba wątków na rdzeń CPU	2
Rozmiar pamięci operacyjnej (RAM)	16 GB
Rodzaj pamięci stałej	NVMe SSD
Rozmiar pamięci stałej	128 GB

is a sample program for testing the IDS.", a następnie kończący się. Program ten również skompilowano przy pomocy środowiska Visual Studio 2019 i kompilatora MSVC. Właściwości programu monitorowanego znajdują się w Tabeli 4.

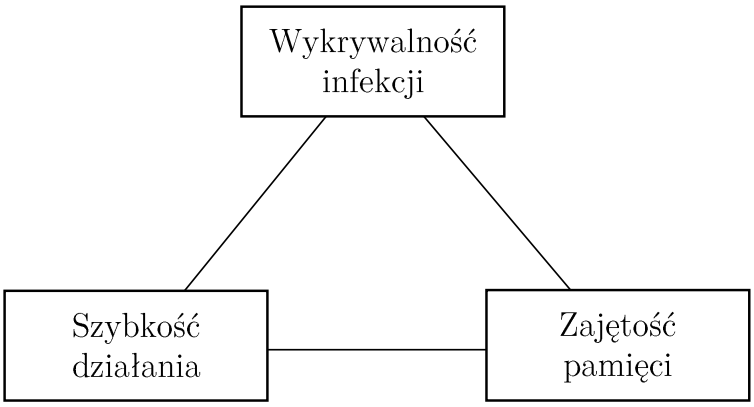
Tabela 4. Właściwości programu monitorowanego przez IDS

Właściwość	**Wartość**
Nazwa pliku	test.exe
Rozmiar pliku	9216 B
Format programu	Win32 Portable Executable (PE)
Platforma docelowa	32-bit

4.1.2 Metodyka i nazewnictwo

Testowanie IDS opiera się na zmianie parametrów wejściowych i sprawdzaniu efektów pracy systemu. Zmiana parametrów wejściowych przekłada się na charakterystykę działania systemu, przez co wyróżnia się kilka kryteriów weryfikacji.

Podstawowymi trzema kryteriami weryfikacji są:



Rysunek 18. Główne kryteria weryfikacji osiągów systemu IDS

- kryterium maksymalizacji wykrywalności - przyjęcie, że istotnym aspektem działania IDS jest uzyskanie maksymalnego wskaźnika wykrywalności anomalii, nawet kosztem zajętości pamięci/prędkości działania,
- kryterium minimalizacji zajętości pamięci - przyjęcie, że istotnym aspektem działania IDS jest uzyskanie niskiej zajętości pamięci stałej lub operacyjnej,
- kryterium maksymalizacji szybkości działania - uznanie możliwie najkrótszego czasu wykonywania algorytmów jako aspektu ważniejszego od dwóch pozostałych.

Na rysunku 18 zobrazowano zależność między kryteriami weryfikacji. Użytkownik IDS może przyjąć, że interesuje go najwyższa możliwa wykrywalność anomalii. Z natury jest tak, że może to spowodować większe zużycie pamięci i/lub wolniejsze działanie systemu. Analogicznie, wysunięcie na pierwszy plan innego kryterium może pogorszyć pozostałe wyniki.

Parametrami wejściowymi systemu są:

- plik monitorowany *test.exe* o rozmiarze F_M [B] - modyfikowany w procesie testowania przez wstrzykiwanie losowych anomalii,
- l - długość pojedynczego receptora podstawowego [b],

- m - próg aktywacji receptorów podstawowych [b],
- l_I - długość pojedynczego receptora pomocniczego (międzykomórkowego) [b],
- m_I - próg aktywacji receptorów pomocniczych (międzykomórkowych) [b].

Parametrami wyjściowymi systemu są:

- TP - ang. true positives - liczba poprawnie wykrytych anomalii - algorytm negatywnej selekcji,
- TP_I - ang. true positives intercellular - liczba poprawnie wykrytych anomalii - po włączeniu proponowanej metody,
- FN - ang. false negatives - liczba niepoprawnie niewykrytych anomalii - algorytm negatywnej selekcji,
- FN_I - ang. false negatives intercellular - liczba niepoprawnie niewykrytych anomalii - po włączeniu proponowanej metody,
- $TP\%$ - ang. true positive ratio - wykrywalność anomalii przy pomocy algorytmu negatywnej selekcji,
- $TP_I\%$ - ang. true positve intercellular ratio - wykrywalność anomalii przy pomocy proponowanej metody,
- $G\%$ - ang. gain ratio - poprawa wykrywalności dzięki włączeniu proponowanej metody,
- T_{Gt} - ang. template generation time - czas generacji wzorców [ms],
- R_{Gt} - ang. receptor generation time - czas generacji receptorów podstawowych [ms],
- R_{IGt} - ang. intercellular receptor generation time - czas generacji receptorów międzykomórkowych [ms],
- Gt - ang. generation time - suma T_{Gt}, R_{Gt} i R_{IGt} [ms],
- FSt - ang. file scan time - średni czas skanowania pliku w poszukiwaniu anomalii [ms],

- R_n - ang. number of receptors - liczba receptorów w pamięci po generacji,
- R_{nI} - liczba receptorów międzykomórkowych w pamięci po generacji,
- R_M - ang. receptor memory - zajętość pamięci przez receptory [B],
- R_{MI} - zajętość pamięci przez receptory międzykomórkowe [B],
- R_{MT} - łączna zajętość pamięci przez wszystkie receptory (podstawowe i międzykomórkowe) [B],
- $R_{MT}\%$ - stosunek łącznej zajętości pamięci przez wszystkie receptory do oryginalnego rozmiaru programu.

Parametr $TP\%$ obliczany jest w następujący sposób:

$$TP\% = \frac{TP}{TP + FN}. \tag{13}$$

Parametr $TP_I\%$ obliczany jest w następujący sposób:

$$TP_I\% = \frac{TP_I}{TP_I + FN_I}. \tag{14}$$

Parametr $G\%$ obliczany jest w następujący sposób:

$$G\% = (TP_I - TP) \cdot 100\%. \tag{15}$$

Parametr Gt obliczany jest w następujący sposób:

$$Gt = T_{Gt} + R_{Gt} + R_{IGt}. \tag{16}$$

Parametr R_{MT} obliczany jest w następujący sposób:

$$R_{MT} = R_M + R_{MI}. \tag{17}$$

Parametr $R_{MT}\%$ obliczany jest w następujący sposób:

$$R_{MT}\% = \frac{R_{MT}}{F_M}. \tag{18}$$

Zakresem testowym L długości receptorów l jest:

$$L = \{16, 17, 18, ..., 32\}, l \in L. \tag{19}$$

Zakresem testowym M progu aktywacji m jest:

$$M = \{8, 9, ..., 12\}, m \in M. \tag{20}$$

Długość receptorów międzykomórkowych l_I ustawiono na 16, ponieważ jest to najmniejsza możliwa liczba bitów obejmujących pełne skrajne bajty sąsiadujących komórek $(2 \cdot 8)$. Zakresem testowym progu aktywacji receptorów międzykomórkowych m_I jest:

$$M_I = \{8, 9, 10, 11\}, m_I \in M_I. \tag{21}$$

Przetestowane muszą być wszystkie kombinacje l, m, l_I, m_I, więc liczba testów do wykonania wynosi:

$$|L| \cdot |M| \cdot |M_I| = 17 \cdot 5 \cdot 4 = 340. \tag{22}$$

Unikalna kombinacja parametrów l i m wykorzystana w konkretnym teście nazywana jest wektorem głównym i oznaczana jest przez $W(l, m)$. Przykładowo, mówiąc o wektorze głównym $W(32, 8)$ mówimy o kombinacji $l = 32$, $m = 8$. Wektorem pomocniczym nazywana jest każda unikalna kombinacja parametrów l_I, m_I i oznaczana jest przez $P(l_I, m_I)$. Na przykład, na wektor pomocniczy $P(16, 8)$ składa się kombinacja $l_I = 16$ i $m_I = 8$. Pojedynczym testem nazwiemy przetestowanie IDS dla zadanej kombinacji $W(l, m)$, $P(l_I, m_I)$.

Proces testowania dla pojedynczego testu został przedstawiony poniżej, w formie listy kroków:

1. Zapisz kopię zapasową pliku *test.exe*
2. Wybierz nowy wektor główny $W(l, m)$ i nowy wektor pomocniczy $P(l_I, m_I)$
3. Wygeneruj nowe zbiory receptorów dla pliku *test.exe*: $\mathbf{R}$ dla $W(l, m)$ i $\mathbf{R_I}$ dla $P(l_I, m_I)$
4. Wstrzyknij jedną 4-bajtową anomalię do pliku *test.exe* w losowym miejscu
5. Uruchom jednostkę wykrywania anomalii
6. Przywróć plik *test.exe* z kopii zapasowej
7. Powtórz kroki 4-6 99 więcej razy w celu uzyskania łącznej liczby 100 prób wykrycia
8. Powtórz cały proces od kroku 2 dopóki nie zostaną przetestowane wszystkie możliwe wektory W i P

Zapisywanie kopii zapasowej pliku *test.exe* następuje poprzez skopiowanie go do katalogu C:\idsbackup pod nazwą *testbackup.exe*. Wybór kolejnych wektorów głównych i pomocniczych realizowany jest w taki sposób, by przetestowane zostały wszystkie możliwe kombinacje l, m, l_I, m_I. Do pliku *test.exe* wstrzykiwane są anomalie o rozmiarze 4 B każda. Każda anomalia nadpisuje dane pliku monitorowanego w losowym miejscu. Dla każdej kombinacji W, P wykonano 100 testów wykrywania anomalii. Daje to łącznie $340 \cdot 100 = 34000$ prób wykrycia anomalii.

4.1.3 Wyniki wg kryterium maksymalizacji wykrywalności

Kryterium maksymalizacji wydajności zakłada osiągnięcie jak najwyższego parametru $TP_I\%$. Wyniki przedstawiono w poniższych tabelach. Wyniki badań posortowane są najpierw od najwyższej wartości $TP_I\%$, a w następnej kolejności od najniższych zajętości pamięci, R_{MT}. Tabele z wynikami podzielone są na kilkuprocentowe odstępy wykrywalności, chyba, że wyników byłoby w tabeli zbyt mało - wtedy wykrywalności mają szerszy zakres w tabelach.

Tabela 5. Wyniki wg kryterium maksymalizacji wykrywalności (1-24)

#	**W**	**P**	**TP%**	**TP_I%**	**G%**	**R_{MT}%**
1	23, 11	16, 11	94	100	6	45.1
2	20, 11	16, 11	97	100	3	50.5
3	25, 11	16, 11	97	100	3	51.1
4	16, 12	16, 9	100	100	0	65.3
5	32, 11	16, 9	99	100	1	66
6	16, 12	16, 10	100	100	0	71.7
7	32, 11	16, 10	100	100	0	74.2
8	20, 12	16, 8	100	100	0	78.5
9	22, 12	16, 9	100	100	0	83
10	23, 12	16, 8	100	100	0	84.8
11	23, 12	16, 9	100	100	0	84.8
12	22, 12	16, 10	100	100	0	87.1
13	23, 12	16, 10	100	100	0	87.6
14	21, 12	16, 11	99	100	1	91.3
15	32, 11	16, 11	98	100	2	93.4
16	25, 12	16, 8	100	100	0	94.6
17	25, 12	16, 10	100	100	0	98

Tabela 5. Wyniki wg kryterium maksymalizacji wykrywalności (1-24) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
18	20, 12	16, 11	99	100	1	100.7
19	24, 12	16, 8	100	100	0	102.3
20	22, 12	16, 11	100	100	0	103
21	23, 12	16, 11	100	100	0	103.2
22	24, 12	16, 9	100	100	0	104.2
23	27, 12	16, 9	100	100	0	104.8
24	26, 12	16, 10	100	100	0	108

Przy wyłączonej metodzie ICR, testy 1-24 wykazały wykrywalność na poziomie od 94% do 100%. Po włączeniu metody ICR, wszystkie 24 testy wykazały wykrywalność na poziomie 100%. Można zaobserwować, że dzięki metodzie ICR osiągnięto wzrost wykrywalności w zakresie 1-6 punktów procentowych. Należy odnotować, że zajętość pamięci przez receptory od testu 18 wzwyż osiągnęła 100% względem oryginalnego rozmiaru monitorowanego programu. Oznacza to nieopłacalność zastosowania wektorów W, P powyżej testu 18 dla tego monitorowanego programu.

Tabela 6. Wyniki wg kryterium maksymalizacji wykrywalności (25-42)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
25	24, 12	16, 10	100	100	0	111.3
26	29, 12	16, 8	100	100	0	115.5
27	28, 12	16, 8	100	100	0	118.7
28	28, 12	16, 9	100	100	0	119.9
29	29, 12	16, 10	100	100	0	120
30	26, 12	16, 11	100	100	0	125

Tabela 6. Wyniki wg kryterium maksymalizacji wykrywalności (25-42) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
31	30, 12	16, 9	100	100	0	125.3
32	31, 12	16, 8	100	100	0	125.6
33	27, 12	16, 11	100	100	0	126
34	28, 12	16, 10	100	100	0	126.4
35	24, 12	16, 11	100	100	0	129.1
36	29, 12	16, 11	99	100	1	137.7
37	28, 12	16, 11	100	100	0	144.7
38	32, 12	16, 8	100	100	0	147.4
39	31, 12	16, 11	100	100	0	148.8
40	30, 12	16, 11	99	100	1	148.8
41	32, 12	16, 9	99	100	1	150.3
42	32, 12	16, 11	98	100	2	177.8

Dla testów 25-42, wykrywalność osiągnęła w większości 100%, jednak zajętość pamięci przez receptory wyniosła od 111% F_M do 178% F_M. Taka zajętość pamięci oznacza nieopłacalność wykorzystania wektorów $Wi\ P$ dla monitorowanego pliku. Można przypuszczać, że wysoka zajętość pamięci ma związek z wartościami parametrów w wektorze W. W czterech przypadkach (testy 36, 40, 41, 42) metoda ICR pozwoliła na poprawienie wydajności o od 1 do 2 punktów procentowych.

Tabela 7. Wyniki wg kryterium maksymalizacji wykrywalności (43-65)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
43	16, 10	16, 11	85	99	14	32.6
44	28, 10	16, 11	94	99	5	40
45	27, 11	16, 10	95	99	4	40.4

Tabela 7. Wyniki wg kryterium maksymalizacji wykrywalności (43-65) (cd.)

#	**W**	**P**	**TP%**	**TP_I%**	**G%**	**R_{MT}%**
46	24, 11	16, 8	99	99	0	40.8
47	29, 11	16, 10	99	99	0	44.5
48	22, 11	16, 11	95	99	4	48.8
49	32, 10	16, 11	92	99	7	54.8
50	29, 11	16, 11	97	99	2	62.2
51	32, 11	16, 8	99	99	0	63.1
52	19, 12	16, 8	99	99	0	64.4
53	19, 12	16, 10	99	99	0	65.7
54	18, 12	16, 10	99	99	0	66.6
55	17, 12	16, 11	98	99	1	68.9
56	30, 11	16, 11	94	99	5	70.9
57	21, 12	16, 8	99	99	0	73.9
58	21, 12	16, 9	99	99	0	73.9
59	21, 12	16, 10	99	99	0	76.2
60	20, 12	16, 9	99	99	0	79
61	18, 12	16, 11	99	99	0	81
62	22, 12	16, 8	99	99	0	82.8
63	20, 12	16, 10	99	99	0	84.2
64	16, 12	16, 11	99	99	0	87.8
65	25, 12	16, 9	99	99	0	94.7

Testy 43-65 wykazały wykrywalność na poziomie 99%. Zastosowanie metody ICR pozwoliło w ośmiu przypadkach na poprawę wykrywalności o od 1 do 14 punktów procentowych. Test 43 pokazał, że system osiągnął 99% wykrywalności przy zajętości pamięci poniżej $\frac{1}{3}$ F_M. Wynik ten możliwy był dzięki załącze-

niu metody ICR, która poprawiła wykrywalność aż o 14 punktów procentowych. Warto także zauważyć, że wszystkie z tych testów pozwoliły na zaoszczędzenie pamięci.

Tabela 8. Wyniki wg kryterium maksymalizacji wykrywalności (66-73)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
66	26, 12	16, 8	99	99	0	102.6
67	26, 12	16, 9	99	99	0	103.3
68	27, 12	16, 8	99	99	0	104.5
69	27, 12	16, 10	99	99	0	109.1
70	25, 12	16, 11	99	99	0	114.6
71	29, 12	16, 9	99	99	0	115.6
72	31, 12	16, 9	99	99	0	125.7
73	30, 12	16, 10	99	99	0	131

Testy 66-73 pokazały nieopłacalność wykorzystania odpowiadających im wektorów $Wi\ P$, ponieważ mimo wykrywalności 99%, w każdym przypadku zajętość pamięci wyniosła ponad 100% F_M.

Tabela 9. Wyniki wg kryterium maksymalizacji wykrywalności (74-97)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
74	22, 10	16, 11	69	98	29	26.2
75	22, 11	16, 10	98	98	0	32.9
76	20, 11	16, 10	97	98	1	34
77	26, 11	16, 9	98	98	0	37.5
78	18, 11	16, 11	93	98	5	37.7
79	21, 11	16, 11	94	98	4	39.6
80	26, 11	16, 10	96	98	2	42.2

Tabela 9. Wyniki wg kryterium maksymalizacji wykrywalności (74-97) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
81	24, 11	16, 9	97	98	1	42.6
82	28, 11	16, 8	98	98	0	46.1
83	31, 11	16, 10	97	98	1	49.8
84	30, 11	16, 10	98	98	0	53.1
85	28, 11	16, 10	95	98	3	53.9
86	27, 11	16, 11	94	98	4	57.3
87	26, 11	16, 11	96	98	2	59.2
88	18, 12	16, 8	98	98	0	63.4
89	18, 12	16, 9	98	98	0	63.8
90	19, 12	16, 9	98	98	0	64.4
91	24, 11	16, 11	96	98	2	67.5
92	31, 11	16, 11	95	98	3	67.6
93	28, 11	16, 11	97	98	1	72.2
94	19, 12	16, 11	96	98	2	79.4
95	30, 12	16, 8	98	98	0	124.5
96	31, 12	16, 10	97	98	1	131.1
97	32, 12	16, 10	98	98	0	158.5

Dla poziomu wykrywalności 98% można zaobserwować, że w testach 74-97 wystąpiły bardzo korzystne zajętości pamięci. W przypadku $W(22, 10)$ $P(16, 11)$ (test 74) zajętość pamięci wyniosła jedynie 26.2% F_M. Zastosowanie metody ICR dało w tym przypadku wzrost wykrywalności aż o 29 punktów procentowych. Łatwo zauważyć, że tylko w 9 na 24 przypadków metoda ICR nie przyniosła poprawy. Testy 95-97 wykazały nieopłacalność zastosowania wektorów $W(30, 12)P(16, 8)$, $W(31, 12)P(16, 10)$ i $W(32, 12)P(16, 10)$ ze względu na zaję-

tość pamięci.

Tabela 10. Wyniki wg kryterium maksymalizacji wykrywalności (98-112)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
98	16, 11	16, 8	97	97	0	24.6
99	16, 11	16, 9	97	97	0	26.2
100	20, 11	16, 9	97	97	0	28.8
101	23, 11	16, 10	94	97	3	29.5
102	20, 10	16, 11	82	97	15	29.9
103	24, 9	16, 11	61	97	36	30.1
104	16, 11	16, 10	96	97	1	32.6
105	19, 11	16, 11	92	97	5	32.6
106	27, 11	16, 9	97	97	0	36.1
107	26, 11	16, 8	97	97	0	36.8
108	24, 10	16, 11	88	97	9	40.8
109	31, 11	16, 8	97	97	0	44.4
110	24, 11	16, 10	97	97	0	49.7
111	17, 12	16, 9	97	97	0	54.9
112	16, 12	16, 8	97	97	0	63.7

Wyniki testów 98-112 pokazały wykrywalność na poziomie 97%. Warto odnotować w tym miejscu bardzo wysoką poprawę wykrywalności (61% $\rightarrow$ 97%) dzięki metodzie ICR w przypadku testu 103. Jest to znakomita poprawa biorąc pod uwagę, że jest to jedyny w tabeli test z parametrem $m = 9$. Najwyższa zajętość pamięci dla tej wykrywalności wyniosła 63.7% F_M. Mediana wartości m wyniosła dla tej części testów 11.

Tabela 11. Wyniki wg kryterium maksymalizacji wykrywalności (113-124)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
113	23, 11	16, 8	96	96	0	26.6
114	20, 11	16, 8	96	96	0	28.3
115	22, 11	16, 9	96	96	0	28.8
116	25, 11	16, 9	96	96	0	31.2
117	25, 11	16, 10	96	96	0	34.5
118	32, 10	16, 10	93	96	3	35.6
119	27, 11	16, 8	96	96	0	35.9
120	29, 11	16, 8	96	96	0	40
121	29, 11	16, 9	96	96	0	40.1
122	30, 11	16, 8	96	96	0	46.5
123	30, 11	16, 9	96	96	0	47.3
124	28, 11	16, 9	95	96	1	47.3

Dla powyższych wyników można zaobserwować wykrywalność na poziomie 96%. Zajętości pamięci są atrakcyjne, a w przypadku testu 118 metoda ICR pomogła uzyskać wykrywalność o trzy punkty procentowe wyższą niż w przypadku metody niezmodyfikowanej.

Tabela 12. Wyniki wg kryterium maksymalizacji wykrywalności (125-148)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
125	20, 10	16, 10	86	95	9	13.3
126	21, 11	16, 10	91	95	4	24.6
127	32, 10	16, 9	93	95	2	27.4
128	17, 12	16, 8	95	95	0	54.9
129	17, 12	16, 10	95	95	0	56.1

Tabela 12. Wyniki wg kryterium maksymalizacji wykrywalności (125-148) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
130	28, 10	16, 10	89	94	5	21.8
131	32, 10	16, 8	94	94	0	24.5
132	31, 10	16, 11	78	94	16	32.9
133	30, 10	16, 11	91	94	3	37.3
134	18, 10	16, 11	51	93	42	21.4
135	16, 8	16, 11	20	93	73	24.6
136	16, 9	16, 11	53	93	40	26.2
137	27, 10	16, 11	64	93	29	28.5
138	22, 11	16, 8	93	93	0	28.6
139	29, 10	16, 11	69	93	24	29.7
140	25, 11	16, 8	93	93	0	31.1
141	21, 11	16, 9	91	92	1	22.3
142	20, 8	16, 11	0	92	92	22.3
143	24, 10	16, 10	91	92	1	22.9
144	20, 9	16, 11	26	92	66	23
145	18, 11	16, 10	91	92	1	23.4
146	25, 10	16, 11	55	92	37	24.4
147	23, 11	16, 9	92	92	0	26.7
148	26, 10	16, 11	80	92	12	31

Wyniki testów 125-148 pokazują, że postępuje stopniowy spadek poziomów wykrywalności. Poziom 95% został osiągnięty tylko przez 5 testów, poziom 94% przez 4 testy, poziom 93% przez 7 testów, a poziom 92% przez 8 testów. Metoda ICR osiągnęła w tym przedziale testów bardzo dobre wyniki, poprawiając oryginalne wykrywalności aż w 18 przypadkach. Poprawa wyniosła od 1 do aż

92 punktów procentowych, w przypadku testu 142. Warto zauważyć, że zajętości pamięci są na ogół niskie, a rekordowo niską zajętość pamięci na poziomie 13.3% F_M osiągnięto stosując metodę ICR, która poprawiła oryginalną wykrywalność o 9 punktów procentowych.

Tabela 13. Wyniki wg kryterium maksymalizacji wykrywalności (149-173)

#	**W**	**P**	**TP%**	**TP_I%**	**G%**	**R_{MT}%**
149	19, 11	16, 8	91	91	0	17.7
150	22, 8	16, 11	0	91	91	20.2
151	18, 11	16, 9	91	91	0	20.5
152	32, 8	16, 11	40	91	51	32.4
153	16, 11	16, 11	91	91	0	48.7
154	28, 10	16, 9	87	90	3	15.2
155	30, 10	16, 10	85	90	5	19.5
156	24, 8	16, 11	13	90	77	27.3
157	28, 9	16, 11	56	90	34	28.9
158	31, 11	16, 9	90	90	0	44.4
159	28, 9	16, 10	59	89	30	10.6
160	24, 10	16, 9	88	89	1	15.8
161	23, 9	16, 11	0	89	89	18.4
162	21, 10	16, 11	40	89	49	20.2
163	27, 9	16, 11	0	89	89	21.5
164	17, 11	16, 11	86	89	3	29
165	16, 9	16, 10	58	88	30	10.1
166	32, 9	16, 9	79	88	9	11.1
167	28, 10	16, 8	88	88	0	14
168	17, 11	16, 8	88	88	0	14.9

Tabela 13. Wyniki wg kryterium maksymalizacji wykrywalności (149-173) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
169	19, 11	16, 9	88	88	0	17.7
170	23, 8	16, 11	0	88	88	18.4
171	18, 11	16, 8	88	88	0	20.1
172	26, 9	16, 11	21	88	67	23.3
173	28, 8	16, 11	8	88	80	26.4

W przypadku testów 149-173, wykrywalności wahały się między 91% a 88%. Załączenie metody ICR dawało zwykle bardzo dobre rezultaty z niskimi zajętościami pamięci przez receptory. Ponieważ wykrywalność spadła już do poziomu 88%, te wektory testowe mogą nie wystarczyć, zależnie od obranego poziomu wykrywalności satysfakcjonującego kryterium.

Tabela 14. Wyniki wg kryterium maksymalizacji wykrywalności (174-195)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
174	16, 8	16, 10	20	87	67	8.5
175	24, 9	16, 10	63	87	24	12.2
176	30, 10	16, 9	86	87	1	13.7
177	17, 11	16, 9	87	87	0	14.9
178	17, 10	16, 11	27	87	60	14.9
179	22, 9	16, 11	4	87	83	20.3
180	21, 11	16, 8	87	87	0	22.3
181	31, 8	16, 11	0	87	87	23.2
182	30, 9	16, 11	24	87	63	25.8
183	32, 9	16, 11	63	87	24	38.6
184	32, 9	16, 8	86	86	0	8.2

Tabela 14. Wyniki wg kryterium maksymalizacji wykrywalności (174-195) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
185	16, 10	16, 9	84	86	2	10.1
186	30, 10	16, 8	86	86	0	12.9
187	24, 10	16, 8	86	86	0	14
188	19, 8	16, 11	0	85	85	15
189	17, 11	16, 10	85	85	0	16.1
190	18, 8	16, 11	0	85	85	17.6
191	19, 11	16, 10	83	85	2	18.9
192	32, 9	16, 10	70	85	15	19.3
193	23, 10	16, 11	24	85	61	20.9
194	27, 8	16, 11	0	85	85	21.5
195	26, 8	16, 11	0	85	85	22.4

Testy od 173 do 195 zademonstrowały wykrywalności na poziomach pomiędzy 87% a 85%. W przypadku kombinacji $W(16,8)P(16,10)$ widać, że zajętość pamięci była na bardzo korzystnym poziomie 8.5% przy znakomitym wzroście wykrywalności o 67 punktów procentowych.

Tabela 15. Wyniki wg kryterium maksymalizacji wykrywalności (196-219)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
196	16, 10	16, 8	84	84	0	8.5
197	29, 10	16, 10	74	84	10	12
198	32, 8	16, 10	34	84	50	13.2
199	17, 8	16, 11	0	84	84	14
200	21, 9	16, 11	0	84	84	17.4
201	29, 8	16, 11	0	84	84	22.2

Tabela 15. Wyniki wg kryterium maksymalizacji wykrywalności (196-219) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
202	20, 9	16, 10	36	83	47	6.4
203	25, 9	16, 11	0	83	83	19.9
204	29, 9	16, 11	0	83	83	22.2
205	20, 10	16, 8	82	82	0	7.6
206	26, 10	16, 10	74	82	8	14
207	31, 10	16, 10	76	82	6	15.1
208	24, 9	16, 8	78	81	3	3.3
209	31, 10	16, 8	81	81	0	9.6
210	21, 8	16, 11	0	81	81	17.4
211	20, 10	16, 9	79	80	1	8.1
212	17, 9	16, 11	0	80	80	14
213	19, 9	16, 11	0	80	80	15
214	16, 10	16, 10	80	80	0	16.5
215	18, 9	16, 11	6	79	73	17.9
216	27, 10	16, 8	77	77	0	7
217	29, 10	16, 9	76	77	1	7.6
218	27, 10	16, 10	65	77	12	11.6
219	31, 9	16, 11	0	77	77	23.2

Na podstawie powyższych testów widać wyraźnie, że dla tych wartości wektorów W i P wykrywalności spadały. Załączenie metody ICR dało w ośmiu przypadkach wzrost wykrywalności aż o 80 punktów procentowych. Dla $W(24, 9)P(16, 8)$ przy TP_I na poziomie już tylko 78%, zajętość pamięci wyniosła tylko 3.3% F_M. W zależności od administratora systemu, te wykrywalności te mogą już nie wystarczyć.

Tabela 16. Wyniki wg kryterium maksymalizacji wykrywalności (220-243)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
220	24, 9	16, 9	69	76	7	5.1
221	24, 8	16, 10	11	76	65	9.5
222	31, 10	16, 9	76	76	0	9.7
223	30, 8	16, 11	0	76	76	24.3
224	26, 10	16, 9	73	75	2	9.2
225	19, 10	16, 11	21	75	54	16.1
226	25, 8	16, 11	0	75	75	19.9
227	20, 8	16, 10	0	72	72	5.7
228	26, 9	16, 10	22	72	50	6.3
229	28, 8	16, 10	6	72	66	8.1
230	22, 10	16, 9	69	71	2	6.1
231	27, 10	16, 9	70	71	1	7.2
232	29, 10	16, 8	70	70	0	7.5
233	26, 10	16, 8	70	70	0	8.5
234	22, 10	16, 10	62	70	8	10.3
235	32, 8	16, 9	26	68	42	5
236	18, 10	16, 10	55	68	13	7
237	16, 9	16, 9	67	67	0	3.7
238	25, 10	16, 10	59	67	8	7.8
239	21, 10	16, 10	39	66	27	5.1
240	30, 9	16, 10	29	65	36	8
241	22, 10	16, 8	64	64	0	6
242	30, 8	16, 10	0	64	64	6.6
243	18, 10	16, 8	63	63	0	3.8

Patrząc na wyniki uzyskane w testach 220-243, nie sposób nie zauważyć, że dla wykrywalności w granicach 76-63% zajętości pamięci to teraz w większości wyniki jednocyfrowe, pomijając część ułamkową. Rozpiętość wzrostu wykrywalności była szeroka, pomiędzy 0 a 76, co dało odchylenie standardowe na poziomie 29.

Tabela 17. Wyniki wg kryterium maksymalizacji wykrywalności (244-267)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
244	16, 8	16, 9	24	61	37	2.1
245	16, 9	16, 8	59	60	1	2.1
246	24, 8	16, 9	19	59	40	2.4
247	18, 10	16, 9	54	58	4	4.1
248	29, 9	16, 10	0	58	58	4.5
249	31, 9	16, 10	0	58	58	5.5
250	23, 10	16, 10	33	56	23	5.3
251	18, 9	16, 10	13	55	42	3.5
252	21, 10	16, 9	53	54	1	2.8
253	28, 9	16, 9	43	54	11	4.1
254	22, 8	16, 10	0	54	54	4.3
255	25, 10	16, 9	54	54	0	4.4
256	27, 8	16, 10	0	54	54	4.6
257	26, 8	16, 10	0	54	54	5.4
258	31, 8	16, 10	0	54	54	5.5
259	28, 9	16, 8	50	51	1	2.9
260	25, 10	16, 8	51	51	0	4.4
261	22, 9	16, 10	6	51	45	4.4
262	29, 8	16, 10	0	50	50	4.5

Tabela 17. Wyniki wg kryterium maksymalizacji wykrywalności (244-267) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
263	21, 9	16, 10	0	49	49	2.3
264	27, 9	16, 10	0	48	48	4.6
265	32, 8	16, 8	34	46	12	2.1
266	21, 10	16, 8	44	44	0	2.8
267	20, 9	16, 9	32	43	11	1.2

W przypadku wyników uzyskanych w testach 244-267, wykrywalności wynosiły już tylko między 61% a 43%. Można zaobserwować wyraźny spadek wykrywalności dla parametrów m i m_I o wartościach równych 8 lub 9. Średnia zajętość pamięci w tych testach wyniosła 3.74%, co jest wynikiem niskim, ale i tak już nieadekwatnym do zbyt niskiej wykrywalności.

Tabela 18. Wyniki wg kryterium maksymalizacji wykrywalności (268-291)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
268	21, 8	16, 10	0	42	42	2.3
269	23, 10	16, 9	41	42	1	2.5
270	25, 9	16, 10	0	42	42	3.3
271	23, 8	16, 10	0	41	41	2.8
272	18, 8	16, 10	0	41	41	3.2
273	23, 10	16, 8	39	39	0	2.4
274	25, 8	16, 10	0	39	39	3.3
275	17, 10	16, 10	27	38	11	2.1
276	30, 9	16, 9	24	37	13	2.2
277	19, 10	16, 10	18	37	19	2.5
278	23, 9	16, 10	0	36	36	2.8

Tabela 18. Wyniki wg kryterium maksymalizacji wykrywalności (268-291) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
279	28, 8	16, 9	6	30	24	1.5
280	17, 8	16, 10	0	27	27	1.2
281	19, 8	16, 10	0	27	27	1.3
282	30, 9	16, 8	27	27	0	1.4
283	16, 8	16, 8	25	25	0	0.5
284	19, 10	16, 9	25	25	0	1.2
285	26, 9	16, 9	13	25	12	1.5
286	20, 9	16, 8	24	24	0	0.7
287	26, 9	16, 8	24	24	0	0.8
288	24, 8	16, 8	12	22	10	0.5
289	17, 10	16, 8	22	22	0	0.9
290	17, 9	16, 10	0	22	22	1.2
291	20, 8	16, 9	0	21	21	0.5

Rekordy osiągów 268-291 pokazują istotny wzrost wykrywalności dzięki zastosowaniu metody ICR, jednak spodziewane jest, że tak jak poprzednio, wykrywalności na poziomach między 42% a 21% są już raczej nie do przyjęcia i nierelewantne.

Tabela 19. Wyniki wg kryterium maksymalizacji wykrywalności (292-313)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
292	17, 10	16, 9	21	21	0	0.9
293	18, 9	16, 9	13	20	7	0.6
294	26, 8	16, 9	0	20	20	0.7
295	19, 9	16, 10	0	19	19	1.3

Tabela 19. Wyniki wg kryterium maksymalizacji wykrywalności (292-313) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
296	22, 9	16, 9	4	17	13	0.2
297	30, 8	16, 9	0	17	17	0.8
298	19, 10	16, 8	16	16	0	1.2
299	28, 8	16, 8	7	15	8	0.3
300	18, 8	16, 9	0	15	15	0.4
301	18, 9	16, 8	14	14	0	0.3
302	22, 8	16, 9	0	13	13	0.2
303	27, 8	16, 9	0	8	8	0.2
304	27, 9	16, 9	0	7	7	0.2
305	29, 8	16, 9	0	6	6	0.1
306	18, 8	16, 8	0	4	4	0
307	29, 9	16, 9	0	4	4	0.1
308	25, 9	16, 9	0	3	3	0
309	31, 8	16, 9	0	3	3	0.1
310	21, 9	16, 9	0	2	2	0
311	25, 8	16, 9	0	2	2	0
312	22, 9	16, 8	2	2	0	0.1
313	23, 8	16, 9	0	1	1	0

W tym przypadku na podstawie powyższych wyników można zauważyć, że wykrywalność spadła już od 21 tylko do jednego procenta. Można zauważyć, że większość parametrów l jest nieparzysta, a parametr m jest bardzo niski i wynosi 8 lub 9. Jest to istotne spostrzeżenie, które pozwala na pewne wnioski co do parametru l.

Tabela 20. Wyniki wg kryterium maksymalizacji wykrywalności (314-340)

#	**W**	**P**	**TP%**	**TP_I%**	**G%**	**R_{MT}%**
314	17, 8	16, 8	0	0	0	0
315	17, 9	16, 8	0	0	0	0
316	19, 8	16, 8	0	0	0	0
317	19, 9	16, 8	0	0	0	0
318	20, 8	16, 8	0	0	0	0
319	21, 8	16, 8	0	0	0	0
320	21, 9	16, 8	0	0	0	0
321	22, 8	16, 8	0	0	0	0
322	23, 8	16, 8	0	0	0	0
323	23, 9	16, 8	0	0	0	0
324	25, 8	16, 8	0	0	0	0
325	25, 9	16, 8	0	0	0	0
326	26, 8	16, 8	0	0	0	0
327	27, 8	16, 8	0	0	0	0
328	27, 9	16, 8	0	0	0	0
329	29, 8	16, 8	0	0	0	0
330	29, 9	16, 8	0	0	0	0
331	30, 8	16, 8	0	0	0	0
332	31, 8	16, 8	0	0	0	0
333	31, 9	16, 8	0	0	0	0
334	17, 8	16, 9	0	0	0	0
335	17, 9	16, 9	0	0	0	0
336	19, 8	16, 9	0	0	0	0
337	19, 9	16, 9	0	0	0	0

Tabela 20. Wyniki wg kryterium maksymalizacji wykrywalności (314-340) (cd.)

#	**W**	**P**	**TP**%	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_{MT}}$%
338	21, 8	16, 9	0	0	0	0
339	23, 9	16, 9	0	0	0	0
340	31, 9	16, 9	0	0	0	0.1

Ostatnie z testów wykrywalności pokazują dobitnie, że wartość m na poziomie 8 jest zdecydowanie nieadekwatna jeżeli administratorowi systemu zależy na parametrze TP i TP_I. W przypadkach testów 314-340, żaden z receptorów nie spowodował wykrycia anomalii. Znamienna jest tutaj również nieparzystość długości receptorów.

4.1.4 Wyniki wg kryterium minimalizacji zajętości pamięci

Kryterium minimalizacji zajętości pamięci zakłada priorytetyzowanie jak najniższego parametru R_{MT} przy zachowaniu akceptowalnych według użytkownika parametrów wykrywalności. Wyniki przedstawiono w następujących tabelach. Wyniki badań posortowane są najpierw od najniższej wartości R_{MT} przy założeniu, że wykrywalność wyniosła co najmniej 75%. Tabele z wynikami podzielone są na kilkuprocentowe odstępy zajętości pamięci (0-10%, 10-15%, 15-20%, itd.), chyba, że wyników byłoby w tabeli zbyt mało - wtedy zajętości mają szerszy zakres w tabelach.

Tabela 21. Wyniki wg kryterium min. zajętości pamięci (1-14)

#	**W**	**P**	**TP_I%**	**G%**	**R_M**	**R_{MI}**	**R_{MT}**	**R_{MT}%**
1	24, 9	16, 8	81	3	279	26	305	3.3
2	24, 9	16, 9	76	7	279	194	473	5.1
3	20, 9	16, 10	83	47	68	524	592	6.4
4	27, 10	16, 8	77	0	645	0	645	7
5	20, 10	16, 8	82	0	700	0	700	7.6
6	29, 10	16, 9	77	1	693	10	703	7.6
7	20, 10	16, 9	80	1	700	46	746	8.1
8	32, 9	16, 8	86	0	684	72	756	8.2
9	16, 8	16, 10	87	67	24	762	786	8.5
10	16, 10	16, 8	84	0	762	24	786	8.5
11	26, 10	16, 9	75	2	787	64	851	9.2
12	24, 8	16, 10	76	65	24	848	872	9.5
13	31, 10	16, 8	81	0	888	0	888	9.6
14	31, 10	16, 9	76	0	888	6	894	9.7
Średnia			80.07	13.79	530.07	184.00	714.07	7.74

Na podstawie wyników 1-14 można stwierdzić, że jeżeli wymagana jest bardzo niska zajętość pamięci (poniżej 10% F_M) to metoda ICR jest w stanie osiągnąć nawet 67% wykrywalność. Średni wzrost wykrywalności wyniósł dla tego zakresu 13.79%.

Tabela 22. Wyniki wg kryterium min. zajętości pamięci (15-35)

#	**W**	**P**	**TP$_I$%**	**G%**	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$%**
15	16, 9	16, 10	88	30	172	762	934	10.1
16	16, 10	16, 9	86	2	762	172	934	10.1
17	28, 9	16, 10	89	30	252	728	980	10.6
18	32, 9	16, 9	88	9	684	342	1026	11.1
19	27, 10	16, 10	77	12	645	422	1067	11.6
20	29, 10	16, 10	84	10	693	414	1107	12
21	24, 9	16, 10	87	24	279	848	1127	12.2
22	30, 10	16, 8	86	0	1189	0	1189	12.9
23	32, 8	16, 10	84	50	120	1094	1214	13.2
24	20, 10	16, 10	95	9	700	524	1224	13.3
25	30, 10	16, 9	87	1	1189	72	1261	13.7
26	28, 10	16, 8	88	0	1278	14	1292	14
27	24, 10	16, 8	86	0	1266	26	1292	14
28	17, 8	16, 11	84	84	0	1292	1292	14
29	26, 10	16, 10	82	8	787	502	1289	14
30	17, 9	16, 11	80	80	0	1292	1292	14
31	17, 11	16, 8	88	0	1377	0	1377	14.9
32	17, 11	16, 9	87	0	1377	0	1377	14.9
33	17, 10	16, 11	87	60	83	1292	1375	14.9
34	19, 8	16, 11	85	85	0	1378	1378	15

Tabela 22. Wyniki wg kryterium min. zajętości pamięci (15-35) (cd.)

#	W	P	TP_I%	G%	R_M	R_{MI}	R_{MT}	R_{MT}%
35	19, 9	16, 11	80	80	0	1378	1378	15
Średnia			85.62	27.33	612.05	597.71	1209.76	13.12

Jeżeli pożądana jest zajętość pamięci w granicach od 10% do 15% F_M, opłacalne może być zastosowanie parametrów $W(16,9)P(16,10)$ lub $W(32,8)P(16,10)$. Średnia wykrywalność wyniosła w tym przypadku 85.62%, a średni wzrost wykrywalności dzięki receptorom międzykomórkowym wyniósł 27.33 punkty procentowe.

Tabela 23. Wyniki wg kryterium min. zajętości pamięci (36-54)

#	W	P	TP_I%	G%	R_M	R_{MI}	R_{MT}	R_{MT}%
36	31, 10	16, 10	82	6	888	504	1392	15.1
37	28, 10	16, 9	90	3	1278	124	1402	15.2
38	24, 10	16, 9	89	1	1266	194	1460	15.8
39	17, 11	16, 10	85	0	1377	108	1485	16.1
40	19, 10	16, 11	75	54	110	1378	1488	16.1
41	16, 10	16, 10	80	0	762	762	1524	16.5
42	21, 9	16, 11	84	84	0	1600	1600	17.4
43	21, 8	16, 11	81	81	0	1600	1600	17.4
44	18, 8	16, 11	85	85	0	1624	1624	17.6
45	19, 11	16, 8	91	0	1627	0	1627	17.7
46	19, 11	16, 9	88	0	1627	0	1627	17.7
47	18, 9	16, 11	79	73	25	1624	1649	17.9
48	23, 9	16, 11	89	89	0	1698	1698	18.4
49	23, 8	16, 11	88	88	0	1698	1698	18.4

Tabela 23. Wyniki wg kryterium min. zajętości pamięci (36-54) (cd.)

#	**W**	**P**	$\mathbf{TP_I}\%$	**G%**	$\mathbf{R_M}$	$\mathbf{R_{MI}}$	$\mathbf{R_{MT}}$	$\mathbf{R_{MT}}\%$
50	19, 11	16, 10	85	2	1627	118	1745	18.9
51	32, 9	16, 10	85	15	684	1094	1778	19.3
52	30, 10	16, 10	90	5	1189	604	1793	19.5
53	25, 9	16, 11	83	83	0	1838	1838	19.9
54	25, 8	16, 11	75	75	0	1838	1838	19.9
Średnia			84.42	39.16	655.79	968.74	1624.53	17.62

Dla testów 36-54 średnia zajętość pamięci wyniosła 17.62% F_M, a wykrywalność nieznacznie spadła względem poprzednich wyników. Największy parametr $G\%$ uzyskano dla $W(23, 9)P(16, 11)$ dzięki wykorzystaniu małej wartości parametru $m = 9$.

Tabela 24. Wyniki wg kryterium min. zajętości pamięci (55-82)

#	**W**	**P**	$\mathbf{TP_I}\%$	**G%**	$\mathbf{R_M}$	$\mathbf{R_{MI}}$	$\mathbf{R_{MT}}$	$\mathbf{R_{MT}}\%$
55	18, 11	16, 8	88	0	1854	2	1856	20.1
56	22, 8	16, 11	91	91	0	1864	1864	20.2
57	21, 10	16, 11	89	49	258	1600	1858	20.2
58	22, 9	16, 11	87	83	6	1864	1870	20.3
59	18, 11	16, 9	91	0	1854	34	1888	20.5
60	23, 10	16, 11	85	61	225	1698	1923	20.9
61	18, 10	16, 11	93	42	347	1624	1971	21.4
62	27, 9	16, 11	89	89	0	1980	1980	21.5
63	27, 8	16, 11	85	85	0	1980	1980	21.5
64	28, 10	16, 10	94	5	1278	728	2006	21.8
65	29, 8	16, 11	84	84	0	2044	2044	22.2

Tabela 24. Wyniki wg kryterium min. zajętości pamięci (55-82) (cd.)

#	**W**	**P**	**TP$_I$**%	**G**%	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$**%
66	29, 9	16, 11	83	83	0	2044	2044	22.2
67	21, 11	16, 9	92	1	2053	2	2055	22.3
68	20, 8	16, 11	92	92	0	2052	2052	22.3
69	21, 11	16, 8	87	0	2053	0	2053	22.3
70	26, 8	16, 11	85	85	0	2068	2068	22.4
71	24, 10	16, 10	92	1	1266	848	2114	22.9
72	20, 9	16, 11	92	66	68	2052	2120	23
73	31, 8	16, 11	87	87	0	2140	2140	23.2
74	31, 9	16, 11	77	77	0	2140	2140	23.2
75	26, 9	16, 11	88	67	78	2068	2146	23.3
76	18, 11	16, 10	92	1	1854	298	2152	23.4
77	30, 8	16, 11	76	76	0	2244	2244	24.3
78	25, 10	16, 11	92	37	407	1838	2245	24.4
79	32, 10	16, 8	94	0	2184	72	2256	24.5
80	16, 11	16, 8	97	0	2246	24	2270	24.6
81	21, 11	16, 10	95	4	2053	210	2263	24.6
82	16, 8	16, 11	93	73	24	2246	2270	24.6
Średnia			88.93	47.82	718.14	1348.71	2066.86	22.43

W granicach R_{MT}% od 20% do 25% wykrywalność wzrosła względem poprzednich wyników o przeszło 4 punkty procentowe do poziomu 88.93%. Średnia zajętość pamięci wyniosła w tym przypadku 22.43% F_M. Można zauważyć, że średni wzrost wykrywalności wyniósł aż 47.82%, co jest znakomitym wynikiem dla tego pułapu zajętości pamięci.

Tabela 25. Wyniki wg kryterium min. zajętości pamięci (83-101)

#	**W**	**P**	$\mathbf{TP_I}$%	**G**%	$\mathbf{R_M}$	$\mathbf{R_{MI}}$	$\mathbf{R_{MT}}$	$\mathbf{R_{MT}}$%
83	30, 9	16, 11	87	63	132	2244	2376	25.8
84	22, 10	16, 11	98	29	550	1864	2414	26.2
85	16, 11	16, 9	97	0	2246	172	2418	26.2
86	16, 9	16, 11	93	40	172	2246	2418	26.2
87	28, 8	16, 11	88	80	18	2412	2430	26.4
88	23, 11	16, 8	96	0	2456	0	2456	26.6
89	23, 11	16, 9	92	0	2456	2	2458	26.7
90	24, 8	16, 11	90	77	24	2492	2516	27.3
91	32, 10	16, 9	95	2	2184	342	2526	27.4
92	20, 11	16, 8	96	0	2605	0	2605	28.3
93	27, 10	16, 11	93	29	645	1980	2625	28.5
94	22, 11	16, 8	93	0	2635	0	2635	28.6
95	20, 11	16, 9	97	0	2605	46	2651	28.8
96	22, 11	16, 9	96	0	2635	16	2651	28.8
97	28, 9	16, 11	90	34	252	2412	2664	28.9
98	17, 11	16, 11	89	3	1377	1292	2669	29
99	23, 11	16, 10	97	3	2456	260	2716	29.5
100	29, 10	16, 11	93	24	693	2044	2737	29.7
101	20, 10	16, 11	97	15	700	2052	2752	29.9
Średnia			93.53	21.00	1412.68	1151.37	2564.05	27.83

Dla testów 83-101 zajętość pamięci wyniosła między 25.8% a 29.9% F_M. Średnia wykrywalność przekroczyła poziom 93.53%, a wzrost wykrywalności jest nadal na imponującym poziomie 21 punktów procentowych.

Tabela 26. Wyniki wg kryterium min. zajętości pamięci (102-113)

#	W	P	TP_I%	G%	R_M	R_{MI}	R_{MT}	R_{MT}%
102	24, 9	16, 11	97	36	279	2492	2771	30.1
103	26, 10	16, 11	92	12	787	2068	2855	31
104	25, 11	16, 8	93	0	2869	0	2869	31.1
105	25, 11	16, 9	96	0	2869	2	2871	31.2
106	32, 8	16, 11	91	51	120	2870	2990	32.4
107	16, 10	16, 11	99	14	762	2246	3008	32.6
108	16, 11	16, 10	97	1	2246	762	3008	32.6
109	19, 11	16, 11	97	5	1627	1378	3005	32.6
110	22, 11	16, 10	98	0	2635	400	3035	32.9
111	31, 10	16, 11	94	16	888	2140	3028	32.9
112	20, 11	16, 10	98	1	2605	524	3129	34
113	25, 11	16, 10	96	0	2869	308	3177	34.5
Średnia			95.67	11.33	1713.00	1265.83	2978.83	32.33

Dla granic 30%-35% F_M (średnio 32.33% F_M) wykrywalność była już na poziomie 95.67%, co oznacza, że wystarczyła tylko $\frac{1}{3}$ oryginalnej zajętości pamięci by uzyskać bardzo wysoką wykrywalność. Metoda ICR poprawiła średnio wyniki TP% o 11.33 punkty procentowe.

Tabela 27. Wyniki wg kryterium min. zajętości pamięci (114-124)

#	W	P	TP_I%	G%	R_M	R_{MI}	R_{MT}	R_{MT}%
114	32, 10	16, 10	96	3	2184	1094	3278	35.6
115	27, 11	16, 8	96	0	3305	0	3305	35.9
116	27, 11	16, 9	97	0	3305	22	3327	36.1
117	26, 11	16, 8	97	0	3390	0	3390	36.8

Tabela 27. Wyniki wg kryterium min. zajętości pamięci (114-124) (cd.)

#	**W**	**P**	**TP$_I$**%	**G**%	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$**%
118	30, 10	16, 11	94	3	1189	2244	3433	37.3
119	26, 11	16, 9	98	0	3390	64	3454	37.5
120	18, 11	16, 11	98	5	1854	1624	3478	37.7
121	32, 9	16, 11	87	24	684	2870	3554	38.6
122	21, 11	16, 11	98	4	2053	1600	3653	39.6
123	28, 10	16, 11	99	5	1278	2412	3690	40
124	29, 11	16, 8	96	0	3687	0	3687	40
Średnia			96.00	4.00	2392.64	1084.55	3477.18	37.74

Testy 114-124 wykazały, że dla pożądanej zajętości pamięci 35.6%-40% odpowiednimi parametrami mogłyby być parametry $W(32,10)P(16,10)$ i $W(32,9)P(16,11)$. Średnia wykrywalność wyniosła 96%.

Tabela 28. Wyniki wg kryterium min. zajętości pamięci (125-133)

#	**W**	**P**	**TP$_I$**%	**G**%	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$**%
125	29, 11	16, 9	96	0	3687	10	3697	40.1
126	27, 11	16, 10	99	4	3305	422	3727	40.4
127	24, 11	16, 8	99	0	3732	26	3758	40.8
128	24, 10	16, 11	97	9	1266	2492	3758	40.8
129	26, 11	16, 10	98	2	3390	502	3892	42.2
130	24, 11	16, 9	98	1	3732	194	3926	42.6
131	31, 11	16, 8	97	0	4089	0	4089	44.4
132	31, 11	16, 9	90	0	4089	6	4095	44.4
133	29, 11	16, 10	99	0	3687	414	4101	44.5
Średnia			97.00	1.78	3441.89	451.78	3893.67	42.24

Zajętości pamięci dla powyższych testów przekroczyły w tym miejscu pułap 40%, co zależnie od wymagań użytkownika systemu, może być już zbyt wysokim poziomem. Niemniej jednak, wykrywalność dla testów wyniosła średnio 96.78%.

Tabela 29. Wyniki wg kryterium min. zajętości pamięci (134-142)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_M}$	$\mathbf{R_{MI}}$	$\mathbf{R_{MT}}$	$\mathbf{R_{MT}}$%
134	23, 11	16, 11	100	6	2456	1698	4154	45.1
135	28, 11	16, 8	98	0	4239	14	4253	46.1
136	30, 11	16, 8	96	0	4290	0	4290	46.5
137	30, 11	16, 9	96	0	4290	72	4362	47.3
138	28, 11	16, 9	96	1	4239	124	4363	47.3
139	16, 11	16, 11	91	0	2246	2246	4492	48.7
140	22, 11	16, 11	99	4	2635	1864	4499	48.8
141	24, 11	16, 10	97	0	3732	848	4580	49.7
142	31, 11	16, 10	98	1	4089	504	4593	49.8
Średnia			96.78	1.33	3579.56	818.89	4398.44	47.70

Dla zajętości w granicach 50.5%-54.9% wykrywalność osiągnęła bardzo wysoki wynik 98.14%. Można rozważać, czy nie jest to już granica opłacalności zastosowania tego kryterium biorąc pod uwagę zajętość pamięci na poziomie połowy oryginalnego programu.

Tabela 30. Wyniki wg kryterium min. zajętości pamięci (143-149)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_M}$	$\mathbf{R_{MI}}$	$\mathbf{R_{MT}}$	$\mathbf{R_{MT}}$%
143	20, 11	16, 11	100	3	2605	2052	4657	50.5
144	25, 11	16, 11	100	3	2869	1838	4707	51.1
145	30, 11	16, 10	98	0	4290	604	4894	53.1
146	28, 11	16, 10	98	3	4239	728	4967	53.9

Tabela 30. Wyniki wg kryterium min. zajętości pamięci (143-149) (cd.)

#	**W**	**P**	**TP$_I$%**	**G%**	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$%**
147	32, 10	16, 11	99	7	2184	2870	5054	54.8
148	17, 12	16, 9	97	0	5058	0	5058	54.9
149	17, 12	16, 8	95	0	5058	0	5058	54.9
Średnia			98.14	2.29	3757.57	1156.00	4913.57	53.31

Dla testów 143-149 widać wyraźnie, że wykrywalność już nie rośnie, w przeciwieństwie do zajętości pamięci na poziomie średnim 53.31%. Jest to zdecydowanie granica opłacalności zastosowania tego algorytmu w tym scenariuszu.

Tabela 31. Wyniki wg kryterium min. zajętości pamięci (150-166)

#	**W**	**P**	**TP$_I$%**	**G%**	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$%**
150	17, 12	16, 10	95	0	5058	108	5166	56.1
151	27, 11	16, 11	98	4	3305	1980	5285	57.3
152	26, 11	16, 11	98	2	3390	2068	5458	59.2
153	29, 11	16, 11	99	2	3687	2044	5731	62.2
154	32, 11	16, 8	99	0	5740	72	5812	63.1
155	18, 12	16, 8	98	0	5844	2	5846	63.4
156	16, 12	16, 8	97	0	5842	24	5866	63.7
157	18, 12	16, 9	98	0	5844	34	5878	63.8
158	19, 12	16, 8	99	0	5936	0	5936	64.4
159	19, 12	16, 9	98	0	5936	0	5936	64.4
160	16, 12	16, 9	100	0	5842	172	6014	65.3
161	19, 12	16, 10	99	0	5936	118	6054	65.7
162	32, 11	16, 9	100	1	5740	342	6082	66
163	18, 12	16, 10	99	0	5844	298	6142	66.6

Tabela 31. Wyniki wg kryterium min. zajętości pamięci (150-166) (cd.)

#	**W**	**P**	**TP$_I$**%	**G**%	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$**%
164	24, 11	16, 11	98	2	3732	2492	6224	67.5
165	31, 11	16, 11	98	3	4089	2140	6229	67.6
166	17, 12	16, 11	99	1	5058	1292	6350	68.9
Średnia			98.35	0.88	5107.24	775.65	5882.88	63.84

Dla wyników zajętości pamięci pomiędzy 55% a 68.9% osiąga się już tylko marginalną poprawę wykrywalności przy pomocy metody ICR.

Tabela 32. Wyniki wg kryterium min. zajętości pamięci (167-176)

#	**W**	**P**	**TP$_I$**%	**G**%	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$**%
167	30, 11	16, 11	99	5	4290	2244	6534	70.9
168	16, 12	16, 10	100	0	5842	762	6604	71.7
169	28, 11	16, 11	98	1	4239	2412	6651	72.2
170	21, 12	16, 8	99	0	6810	0	6810	73.9
171	21, 12	16, 9	99	0	6810	2	6812	73.9
172	32, 11	16, 10	100	0	5740	1094	6834	74.2
173	21, 12	16, 10	99	0	6810	210	7020	76.2
174	20, 12	16, 8	100	0	7233	0	7233	78.5
175	20, 12	16, 9	99	0	7233	46	7279	79
176	19, 12	16, 11	98	2	5936	1378	7314	79.4
Średnia			99.10	0.80	6094.30	814.80	6909.10	74.99

W zakresie R_{MT} od 70% do 80% dominują bardzo wysokie wartości l i m, co powoduje nieopłacalność zastosowania tych parametrów.

Tabela 33. Wyniki wg kryterium min. zajętości pamięci (177-185)

#	**W**	**P**	**TP$_I$**%	**G**%	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$**%
177	18, 12	16, 11	99	0	5844	1624	7468	81
178	22, 12	16, 8	99	0	7629	0	7629	82.8
179	22, 12	16, 9	100	0	7629	16	7645	83
180	20, 12	16, 10	99	0	7233	524	7757	84.2
181	23, 12	16, 8	100	0	7812	0	7812	84.8
182	23, 12	16, 9	100	0	7812	2	7814	84.8
183	22, 12	16, 10	100	0	7629	400	8029	87.1
184	23, 12	16, 10	100	0	7812	260	8072	87.6
185	16, 12	16, 11	99	0	5842	2246	8088	87.8
Średnia			99.56	0.00	7249.11	563.56	7812.67	84.79

Między 80% a 90% zajętości pamięci nie wystąpiły poprawy wykrywalności, ponieważ jest ona na poziomie 99.56%. Mimo zaoszczędzenia prawie 15% miejsca w pamięci dzięki wykorzystaniu metody ICR, w tym przypadku można byłoby uzyskać lepsze rezultaty dla innych parametrów.

Tabela 34. Wyniki wg kryterium min. zajętości pamięci (186-190)

#	**W**	**P**	**TP$_I$**%	**G**%	**R$_M$**	**R$_{MI}$**	**R$_{MT}$**	**R$_{MT}$**%
186	21, 12	16, 11	100	1	6810	1600	8410	91.3
187	32, 11	16, 11	100	2	5740	2870	8610	93.4
188	25, 12	16, 8	100	0	8722	0	8722	94.6
189	25, 12	16, 9	99	0	8722	2	8724	94.7
190	25, 12	16, 10	100	0	8722	308	9030	98
Średnia			99.80	0.60	7743.20	956.00	8699.20	94.40

Dla 90%-100% F_M wyniki są podobne jak w poprzednim przypadku.

Tabela 35. Wyniki wg kryterium min. zajętości pamięci (191-208)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_M}$	$\mathbf{R_{MI}}$	$\mathbf{R_{MT}}$	$\mathbf{R_{MT}}$%
191	20, 12	16, 11	100	1	7233	2052	9285	100.7
192	24, 12	16, 8	100	0	9405	26	9431	102.3
193	26, 12	16, 8	99	0	9455	0	9455	102.6
194	22, 12	16, 11	100	0	7629	1864	9493	103
195	23, 12	16, 11	100	0	7812	1698	9510	103.2
196	26, 12	16, 9	99	0	9455	64	9519	103.3
197	24, 12	16, 9	100	0	9405	194	9599	104.2
198	27, 12	16, 8	99	0	9633	0	9633	104.5
199	27, 12	16, 9	100	0	9633	22	9655	104.8
200	26, 12	16, 10	100	0	9455	502	9957	108
201	27, 12	16, 10	99	0	9633	422	10055	109.1
202	24, 12	16, 10	100	0	9405	848	10253	111.3
203	25, 12	16, 11	99	0	8722	1838	10560	114.6
204	29, 12	16, 8	100	0	10647	0	10647	115.5
205	29, 12	16, 9	99	0	10647	10	10657	115.6
206	28, 12	16, 8	100	0	10924	14	10938	118.7
207	28, 12	16, 9	100	0	10924	124	11048	119.9
208	29, 12	16, 10	100	0	10647	414	11061	120
Średnia			99.67	0.06	9481.33	560.67	10042.00	108.96

Ostatnie testy dla kryterium minimalizacji zajętości pamięci pokazują, że przekroczona została bariera 100% zajętości oryginalnego programu F_M, co w tym miejscu powoduje nieopłacalność zastosowania metody ICR w tej sytuacji.

4.1.5 Wyniki wg kryterium maksymalizacji szybkości działania

Kryterium maksymalizacji szybkości działania zakłada osiągnięcie jak najszybszej generacji receptorów G_t przy jednoczesnym zachowaniu wykrywalności na poziomie co najmniej 75%. Przyjęto, że czas wykonywania algorytmu generacji nie powinien zająć więcej niż 1 sekundę. Wyniki przedstawiono w kolejnych tabelach. Wyniki badań posortowane są najpierw od najniższej wartości G_t. Tabele z wynikami podzielone są na 100-150 milisekundowe przedziały czasu generacji (0-100 ms, 100-150 ms, 150-200 ms, itd.), chyba, że wyników byłoby w tabeli zbyt mało - wtedy czasy mają szersze przedziały w tabelach.

Tabela 36. Wyniki wg kryterium maks. szybkości działania (1-18)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
1	16, 8	16, 10	87	67	8.5	22	3	27	52
2	25, 8	16, 11	75	75	19.9	24	1	28	53
3	17, 8	16, 11	84	84	14	16	0	42	58
4	29, 8	16, 11	84	84	22.2	29	1	33	63
5	27, 8	16, 11	85	85	21.5	27	1	36	64
6	32, 8	16, 10	84	50	13.2	47	3	15	65
7	19, 8	16, 11	85	85	15	20	0	46	66
8	31, 8	16, 11	87	87	23.2	31	0	36	67
9	26, 8	16, 11	85	85	22.4	28	4	38	70
10	24, 8	16, 10	76	65	9.5	37	4	30	71
11	30, 8	16, 11	76	76	24.3	33	1	40	74
12	21, 8	16, 11	81	81	17.4	23	0	53	76
13	23, 8	16, 11	88	88	18.4	25	0	55	80

Tabela 36. Wyniki wg kryterium maks. szybkości działania (1-18) (cd.)

#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
15	28, 8	16, 11	88	80	26.4	34	2	45	81
14	18, 8	16, 11	85	85	17.6	22	2	57	81
16	22, 8	16, 11	91	91	20.2	26	0	63	89
17	17, 9	16, 11	80	80	14	51	1	41	93
18	16, 9	16, 10	88	30	10.1	61	8	27	96
Średnia			83.8	76.6	17.7	30.9	1.7	39.6	72.2

Dla czasów generacji poniżej 100 ms osiągnięto wysoką wykrywalność na poziomie 83.8% i znakomity średni wzrost wykrywalności na poziomie 76.6%. Oznacza to, że zastosowanie metody ICR w przypadku gdy niezbędny jest szybki czas wykonania programu i zachowanie wysokiej wykrywalności, ma sens.

Tabela 37. Wyniki wg kryterium maks. szybkości działania (19-38)

#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
19	20, 9	16, 10	83	47	6.4	76	6	18	100
20	20, 8	16, 11	92	92	22.3	25	1	75	101
21	19, 9	16, 11	80	80	15	62	2	46	110
22	25, 9	16, 11	83	83	19.9	80	2	31	113
24	32, 8	16, 11	91	51	32.4	46	3	68	117
23	28, 9	16, 10	89	30	10.6	103	5	9	117
25	16, 8	16, 11	93	73	24.6	22	4	92	118
26	24, 9	16, 8	81	3	3.3	102	19	2	123
27	21, 9	16, 11	84	84	17.4	70	1	54	125
28	18, 9	16, 11	79	73	17.9	62	7	59	128
30	29, 9	16, 11	83	83	22.2	94	2	33	129

Tabela 37. Wyniki wg kryterium maks. szybkości działania (19-38) (cd.)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
29	24, 9	16, 9	76	7	5.1	102	18	9	129
31	26, 9	16, 11	88	67	23.3	90	5	38	133
32	23, 9	16, 11	89	89	18.4	78	1	55	134
33	27, 9	16, 11	89	89	21.5	89	12	35	136
34	31, 9	16, 11	77	77	23.2	100	2	35	137
35	32, 9	16, 8	86	0	8.2	128	10	2	140
36	32, 9	16, 9	88	9	11.1	127	11	3	141
37	24, 8	16, 11	90	77	27.3	37	4	104	145
38	30, 9	16, 11	87	63	25.8	104	3	40	147
Średnia			85.4	58.9	17.8	79.9	5.9	40.4	126.2

Dla testów 19-38 średnie czasy generacji wyniosły 126.2 ms. Wykrywalność została osiągnięta na poziomie 85.4%, a poprawa wykrywalności - na poziomie 58.9 punktów procentowych. Jest to nadal znakomity i opłacalny wynik, zwłasza, że zajętość pamięci jest nadal niska (17.8%).

Tabela 38. Wyniki wg kryterium maks. szybkości działania (39-49)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
39	24, 9	16, 10	87	24	12.2	102	18	30	150
41	22, 9	16, 11	87	83	20.3	82	6	64	152
40	32, 9	16, 10	85	15	19.3	126	11	15	152
43	24, 9	16, 11	97	36	30.1	96	5	55	156
42	28, 9	16, 11	90	34	28.9	103	5	48	156
44	20, 9	16, 11	92	66	23	75	7	75	157
45	16, 9	16, 11	93	40	26.2	60	8	94	162

Tabela 38. Wyniki wg kryterium maks. szybkości działania (39-49) (cd.)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
46	16, 10	16, 8	84	0	8.5	142	28	5	175
47	17, 10	16, 11	87	60	14.9	140	9	42	191
48	16, 10	16, 9	86	2	10.1	144	31	17	192
49	16, 10	16, 10	80	0	16.5	142	27	27	196
Średnia			88.0	32.7	19.1	110.2	14.1	42.9	167.2

W przedziale pomiędzy 150 a 200 ms wykrywalność wyniosła średnio 88%. Widać tutaj już lekki spadek wzrostu wykrywalności dla metody ICR. Mimo to parametry l i m są nadal stosunkowo niskie i zajętość pamięci jest dzięki temu niska.

Tabela 39. Wyniki wg kryterium maks. szybkości działania (50-73)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
50	32, 9	16, 11	87	24	38.6	127	11	66	204
51	20, 10	16, 8	82	0	7.6	192	30	0	222
52	20, 10	16, 9	80	1	8.1	193	31	4	228
53	19, 10	16, 11	75	54	16.1	174	11	43	228
54	18, 10	16, 11	93	42	21.4	163	16	56	235
55	20, 10	16, 10	95	9	13.3	191	31	19	241
56	16, 10	16, 11	99	14	32.6	142	27	92	261
57	27, 10	16, 8	77	0	7	247	13	2	262
58	27, 10	16, 10	77	12	11.6	247	12	7	266
60	25, 10	16, 11	92	37	24.4	226	13	28	267
59	26, 10	16, 9	75	2	9.2	242	17	8	267
61	24, 10	16, 8	86	0	14	235	31	2	268

Tabela 39. Wyniki wg kryterium maks. szybkości działania (50-73) (cd.)

#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
62	24, 10	16, 9	89	1	15.8	235	31	3	269
63	21, 10	16, 11	89	49	20.2	202	16	53	271
64	24, 10	16, 10	92	1	22.9	235	31	10	276
65	29, 10	16, 9	77	1	7.6	262	14	1	277
66	29, 10	16, 10	84	10	12	262	13	5	280
67	23, 10	16, 11	85	61	20.9	219	19	56	294
70	27, 10	16, 11	93	29	28.5	246	14	35	295
68	31, 10	16, 8	81	0	9.6	276	19	0	295
69	31, 10	16, 9	76	0	9.7	276	18	1	295
71	20, 10	16, 11	97	15	29.9	193	31	74	298
72	26, 10	16, 11	92	12	31	245	17	37	299
73	30, 10	16, 9	87	1	13.7	275	24	1	300
Średnia			85.8	15.6	17.7	221.0	20.4	25.1	266.6

W przedziale czasów generacji Gt między 200 a 300 ms, można zaobserwować spadek wykrywalności o 2.8 punkty procentowe względem wyników uzyskanych w poprzednich testach. Metoda ICR nadal jest opłacalna dla danych wartości.

Tabela 40. Wyniki wg kryterium maks. szybkości działania (74-94)

#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
74	31, 10	16, 10	82	6	15.1	277	19	5	301
75	30, 10	16, 8	86	0	12.9	277	25	0	302
78	22, 10	16, 11	98	29	26.2	215	26	64	305
77	28, 10	16, 9	90	3	15.2	263	41	1	305
76	28, 10	16, 8	88	0	14	264	40	1	305

Tabela 40. Wyniki wg kryterium maks. szybkości działania (74-94) (cd.)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
79	29, 10	16, 11	93	24	29.7	264	14	33	311
80	28, 10	16, 10	94	5	21.8	263	41	8	312
81	26, 10	16, 10	82	8	14	243	65	6	314
82	24, 10	16, 11	97	9	40.8	235	33	56	324
83	30, 10	16, 10	90	5	19.5	296	24	7	327
84	31, 10	16, 11	94	16	32.9	276	17	37	330
85	30, 10	16, 11	94	3	37.3	275	24	39	338
86	28, 10	16, 11	99	5	40	264	40	46	350
87	32, 10	16, 8	94	0	24.5	301	65	2	368
88	32, 10	16, 9	95	2	27.4	302	66	4	372
90	32, 10	16, 10	96	3	35.6	301	65	15	381
89	17, 11	16, 8	88	0	14.9	331	50	0	381
91	17, 11	16, 9	87	0	14.9	331	50	1	382
92	17, 11	16, 10	85	0	16.1	331	50	8	389
93	16, 11	16, 8	97	0	24.6	295	93	3	391
94	16, 11	16, 9	97	0	26.2	296	91	9	396
Średnia			91.7	5.6	24.0	281.0	44.7	16.4	342.1

Dla testów 74-94 które dały średni czas generacji na poziomie 342.1 ms, wykrywalność osiągnęła poziom 91.7%. Metoda ICR daje w tym przypadku wzrost wykrywalności na poziomie 5.6 punktów procentowych. Średni czas generacji receptorów międzykomórkowych wyniósł w tym przypadku 16.4 ms.

Tabela 41. Wyniki wg kryterium maks. szybkości działania (95-103)

#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
95	16, 11	16, 10	97	1	32.6	295	94	27	416
96	17, 11	16, 11	89	3	29	332	50	42	424
97	32, 10	16, 11	99	7	54.8	301	65	67	433
99	18, 11	16, 9	91	0	20.5	373	81	5	459
98	18, 11	16, 8	88	0	20.1	373	80	6	459
100	18, 11	16, 10	92	1	23.4	373	79	13	465
101	16, 11	16, 11	91	0	48.7	296	92	91	479
102	19, 11	16, 8	91	0	17.7	412	70	1	483
103	19, 11	16, 9	88	0	17.7	413	71	1	485
Średnia			91.8	1.3	29.4	352.0	75.8	28.1	455.9

Dla czasów od 400 do 500 ms wykrywalność wzrosła bardzo nieznacznie, co oznacza, że w tym miejscu może być granica opłacalności dla metody ICR w przypadku tego kryterium. Średni łączny czas generacji wzorców wyniósł 352 ms.

Tabela 42. Wyniki wg kryterium maks. szybkości działania (104-110)

#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
104	18, 11	16, 11	98	5	37.7	374	80	56	510
105	19, 11	16, 10	85	2	18.9	440	69	9	518
106	19, 11	16, 11	97	5	32.6	414	68	44	526
107	21, 11	16, 8	87	0	22.3	477	104	0	581
108	20, 11	16, 8	96	0	28.3	430	155	1	586
109	21, 11	16, 10	95	4	24.6	473	104	10	587
110	20, 11	16, 9	97	0	28.8	430	159	5	594
Średnia			93.6	2.3	27.6	434.0	105.6	17.9	557.4

Dla testów 104-110 wykrywalność wzrosła dzięki metodzie ICR o 2.3 punkty procentowe. Średni czas generacji wszystkich receptorów wyniósł 557.4 ms.

Tabela 43. Wyniki wg kryterium maks. szybkości działania (111-122)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
111	20, 11	16, 10	98	1	34	429	157	19	605
112	21, 11	16, 9	92	1	22.3	505	104	2	611
114	21, 11	16, 11	98	4	39.6	477	104	53	634
113	25, 11	16, 8	93	0	31.1	546	88	0	634
115	25, 11	16, 9	96	0	31.2	553	90	3	646
116	25, 11	16, 10	96	0	34.5	548	91	7	646
117	25, 11	16, 11	100	3	51.1	546	88	28	662
118	20, 11	16, 11	100	3	50.5	433	159	74	666
119	23, 11	16, 8	96	0	26.6	525	147	0	672
120	23, 11	16, 9	92	0	26.7	526	145	2	673
121	22, 11	16, 8	93	0	28.6	502	186	1	689
122	22, 11	16, 9	96	0	28.8	503	185	3	691
Średnia			95.8	1.0	33.8	507.8	128.7	16.0	652.4

W przedziale 600 - 700 ms Gt, wskaźnik TP_I% osiągnął wysoki poziom 95.8. Tak wysoki czas generacji może być jednak już nie do przyjęcia dla użytkownika systemu IDS.

Tabela 44. Wyniki wg kryterium maks. szybkości działania (123-142)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
123	26, 11	16, 8	97	0	36.8	571	127	4	702
124	26, 11	16, 9	98	0	37.5	571	130	4	705
125	22, 11	16, 10	98	0	32.9	503	191	14	708

Tabela 44. Wyniki wg kryterium maks. szybkości działania (123-142) (cd.)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
126	26, 11	16, 10	98	2	42.2	570	130	8	708
127	27, 11	16, 10	99	4	40.4	595	111	5	711
128	23, 11	16, 10	97	3	29.5	559	145	10	714
129	27, 11	16, 8	96	0	35.9	597	116	1	714
130	27, 11	16, 9	97	0	36.1	599	118	2	719
131	24, 11	16, 9	98	1	42.6	516	203	4	723
132	24, 11	16, 8	99	0	40.8	516	205	3	724
133	23, 11	16, 11	100	6	45.1	523	147	55	725
134	26, 11	16, 11	98	2	59.2	572	127	35	734
135	24, 11	16, 10	97	0	49.7	516	206	13	735
136	27, 11	16, 11	98	4	57.3	605	114	32	751
137	22, 11	16, 11	99	4	48.8	500	194	64	758
138	29, 11	16, 8	96	0	40	628	136	0	764
139	24, 11	16, 11	98	2	67.5	517	202	54	773
140	29, 11	16, 9	96	0	40.1	636	138	1	775
141	29, 11	16, 10	99	0	44.5	629	143	4	776
142	29, 11	16, 11	99	2	62.2	630	136	32	798
Średnia			97.9	1.5	44.5	567.7	151.0	17.3	735.9

W przedziale 700 - 800 ms *Gt* wyniki wykrywalności poprawiły się bardzo nieznacznie względem poprzednich. Można spostrzec, że w tym miejscu nie opłacają się już dodatkowe milisekundy poświęcone na generację receptorów.

Tabela 45. Wyniki wg kryterium maks. szybkości działania (143-156)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
143	30, 11	16, 8	96	0	46.5	643	178	0	821
144	30, 11	16, 9	96	0	47.3	642	180	1	823
145	31, 11	16, 9	90	0	44.4	657	170	1	828
147	30, 11	16, 10	98	0	53.1	643	180	6	829
146	31, 11	16, 8	97	0	44.4	656	173	0	829
148	31, 11	16, 10	98	1	49.8	658	173	5	836
149	30, 11	16, 11	99	5	70.9	642	177	40	859
151	16, 12	16, 9	100	0	65.3	551	306	8	865
150	16, 12	16, 8	97	0	63.7	550	309	6	865
152	31, 11	16, 11	98	3	67.6	658	174	37	869
153	28, 11	16, 8	98	0	46.1	602	275	1	878
154	16, 12	16, 10	100	0	71.7	549	307	27	883
155	28, 11	16, 9	96	1	47.3	602	281	1	884
156	28, 11	16, 10	98	3	53.9	601	278	9	888
Średnia			97.2	0.9	55.1	618.1	225.8	10.1	854.1

Dla testów 143-156 wyniki według maksymalizacji szybkości działania były gorsze niż w przypadku przedziału 700 - 800 *Gt*.

Tabela 46. Wyniki wg kryterium maks. szybkości działania (157-162)

#	**W**	**P**	$\mathbf{TP_I}$%	**G%**	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
157	28, 11	16, 11	98	1	72.2	602	278	45	925
158	17, 12	16, 9	97	0	54.9	679	271	1	951
160	16, 12	16, 11	99	0	87.8	551	310	92	953
159	17, 12	16, 8	95	0	54.9	680	273	0	953

Tabela 46. Wyniki wg kryterium maks. szybkości działania (157-162) (cd.)

#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
161	17, 12	16, 10	95	0	56.1	679	271	9	959
162	17, 12	16, 11	99	1	68.9	679	272	42	993
Średnia			97.2	0.3	65.8	645.0	279.2	31.5	955.7

Wyniki uzyskane dla przedziału 900 - 1000 ms były porównywalne z poprzednimi.

Tabela 47. Wyniki wg kryterium maks. szybkości działania (163-182)

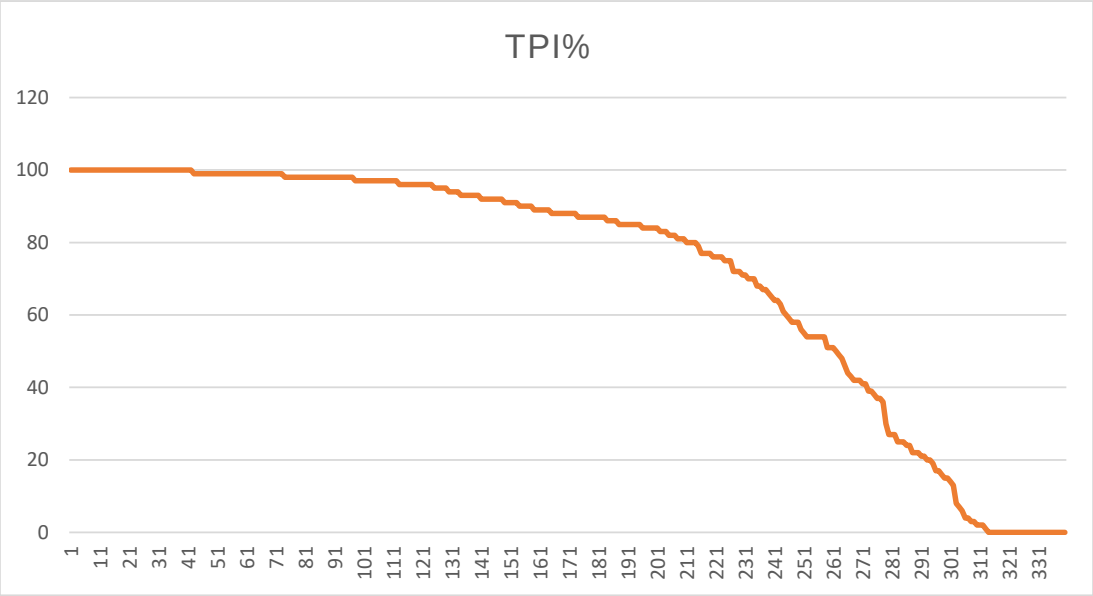
#	W	P	TP_I%	G%	R_{MT}	T_{Gt}	R_{Gt}	R_{IGt}	Gt
163	32, 11	16, 8	99	0	63.1	660	374	1	1035
164	32, 11	16, 9	100	1	66	661	372	4	1037
165	32, 11	16, 10	100	0	74.2	660	369	15	1044
166	32, 11	16, 11	100	2	93.4	659	372	67	1098
167	18, 12	16, 8	98	0	63.4	756	386	6	1148
168	18, 12	16, 10	99	0	66.6	756	385	12	1153
169	18, 12	16, 11	99	0	81	756	389	54	1199
170	18, 12	16, 9	98	0	63.8	817	387	6	1210
171	19, 12	16, 9	98	0	64.4	870	426	1	1297
172	19, 12	16, 8	99	0	64.4	870	427	1	1298
173	19, 12	16, 10	99	0	65.7	869	428	8	1305
174	19, 12	16, 11	98	2	79.4	870	426	45	1341
175	20, 12	16, 8	100	0	78.5	870	698	1	1569
176	20, 12	16, 9	99	0	79	870	696	4	1570
177	20, 12	16, 10	99	0	84.2	869	695	19	1583
178	20, 12	16, 11	100	1	100.7	876	725	75	1676

Tabela 47. Wyniki wg kryterium maks. szybkości działania (163-182) (cd.)

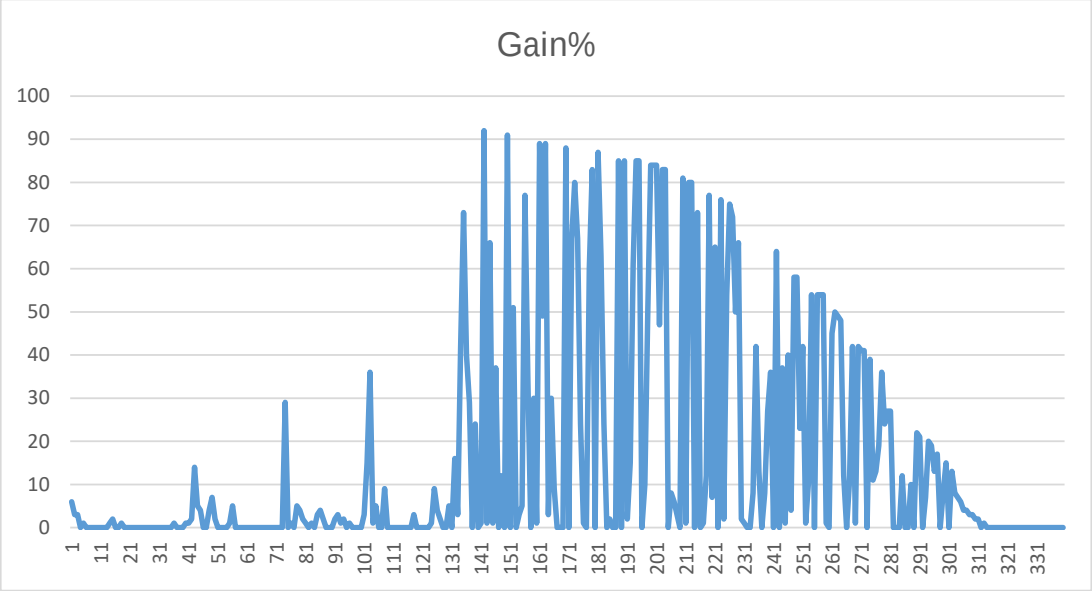
#	**W**	**P**	$\mathbf{TP_I}$%	G%	$\mathbf{R_{MT}}$	$\mathbf{T_{Gt}}$	$\mathbf{R_{Gt}}$	$\mathbf{R_{IGt}}$	**Gt**
179	21, 12	16, 9	99	0	73.9	1017	670	2	1689
180	21, 12	16, 8	99	0	73.9	1018	678	0	1696
181	21, 12	16, 10	99	0	76.2	1015	671	10	1696
182	21, 12	16, 11	100	1	91.3	1016	663	52	1731
Średnia			99.1	0.4	75.2	837.8	511.9	19.2	1368.8

Ostatnie testy dla kryterium maksymalizacji szybkości działania pokazują, że osiągnięcie wykrywalności na poziomie 99% wymaga czasów generacji co najmniej 1 sekundy. Metoda ICR powoduje dla tych czasów nieznaczny już wzrost wykrywalności średnio o 0.4 punkta procentowego.

4.1.6 Podsumowanie testów

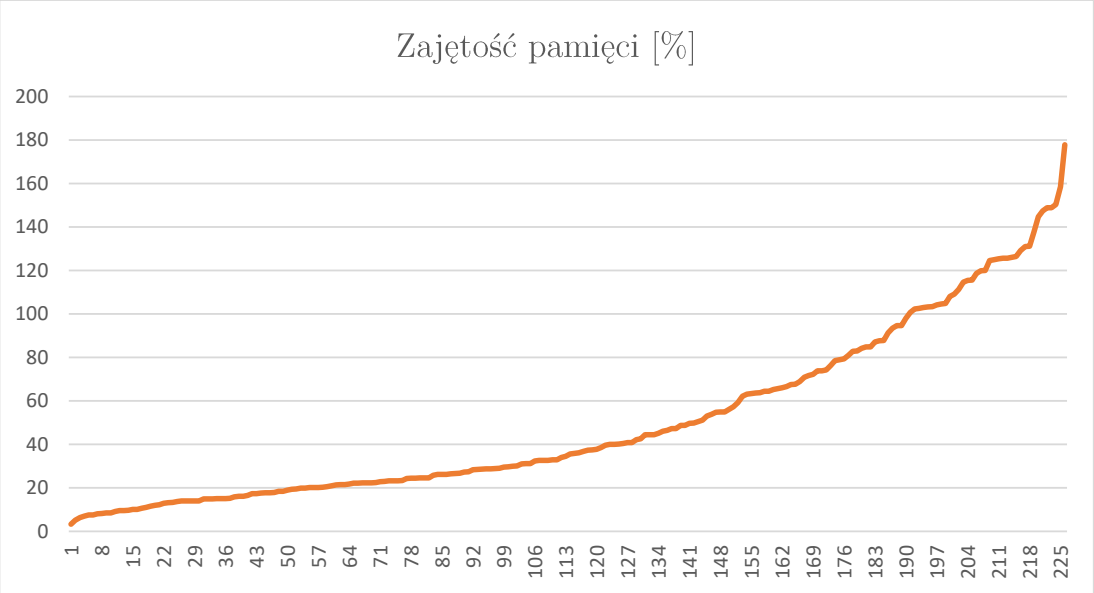


Rysunek 19. Wykres wykrywalności w funkcji numeru testu - kryterium maks. wykrywalności



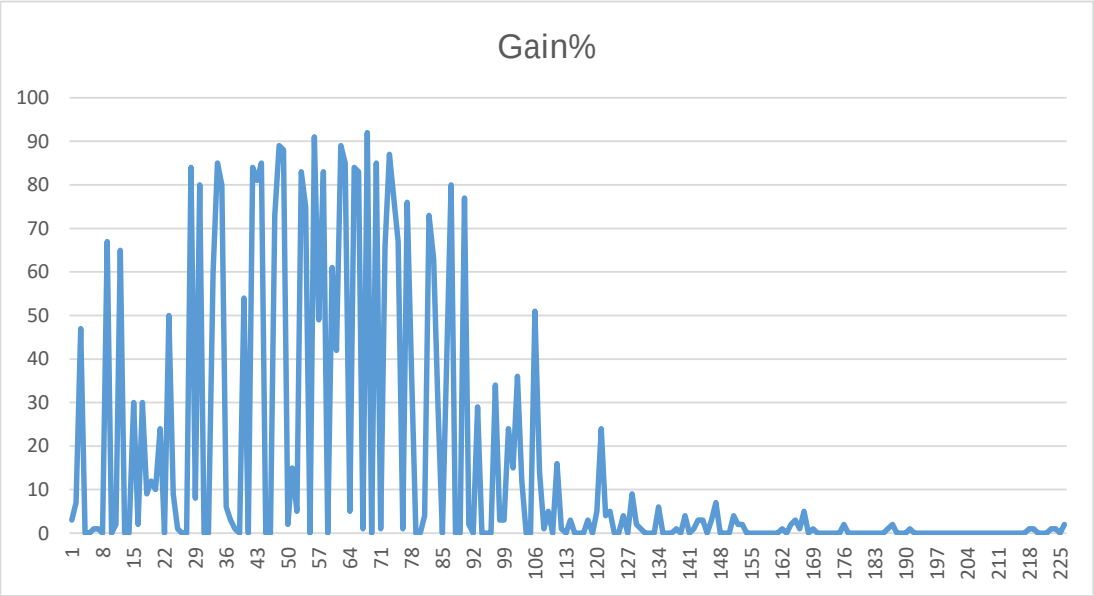
Rysunek 20. Wykres poprawy wykrywalności w funkcji numeru testu - kryterium maks. wykrywalności

Na rysunkach 19 i 20 przedstawiono wyniki testów dla kryterium wykrywalności. Powyższe rozważania sugerują, że najlepsze wyniki metoda ICR osiąga przy wartościach parzystych l i dla wartości m większych lub równych 10. Dla testów 1-3, 42-45, 74, 101-105 można zaobserwować znaczny wzrost wykrywalności, głównie dla wartości l podzielnych przez 4. Metodę ICR opłaca się zastosować dla parzystych wartości parametru l z możliwie najniższym parametrem m i możliwie najwyższym parametrem m_I.



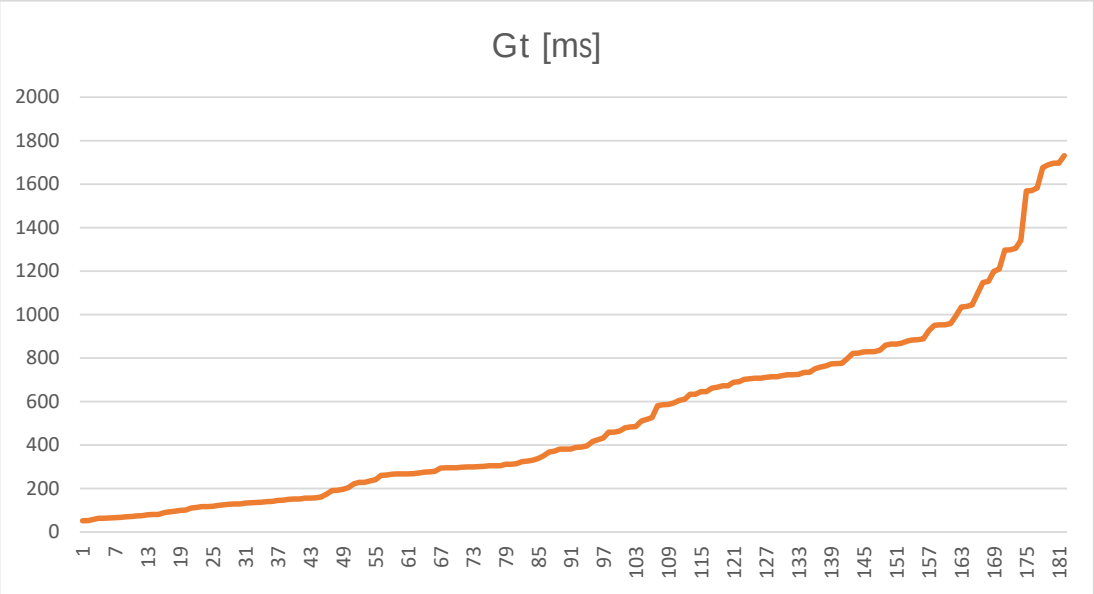
Rysunek 21. Wykres zajętości pamięci w funkcji numeru testu - kryterium min. zajętości pamięci

Na rysunkach 21 i 22 przedstawiono wyniki testów dla kryterium minimalizacji zajętości pamięci. Łatwo zauważyć, że dla tego kryterium metoda ICR świetnie poprawiała wykrywalność nawet dla bardzo niskich zajętości pamięci, co czyni ją atrakcyjną w tym przypadku użycia. Można wywnioskować, że metoda ICR osiągnęła wyniki lepsze od metody klasycznej, przy czym największy wzrost wykrywalności następował w okolicach testu 57.

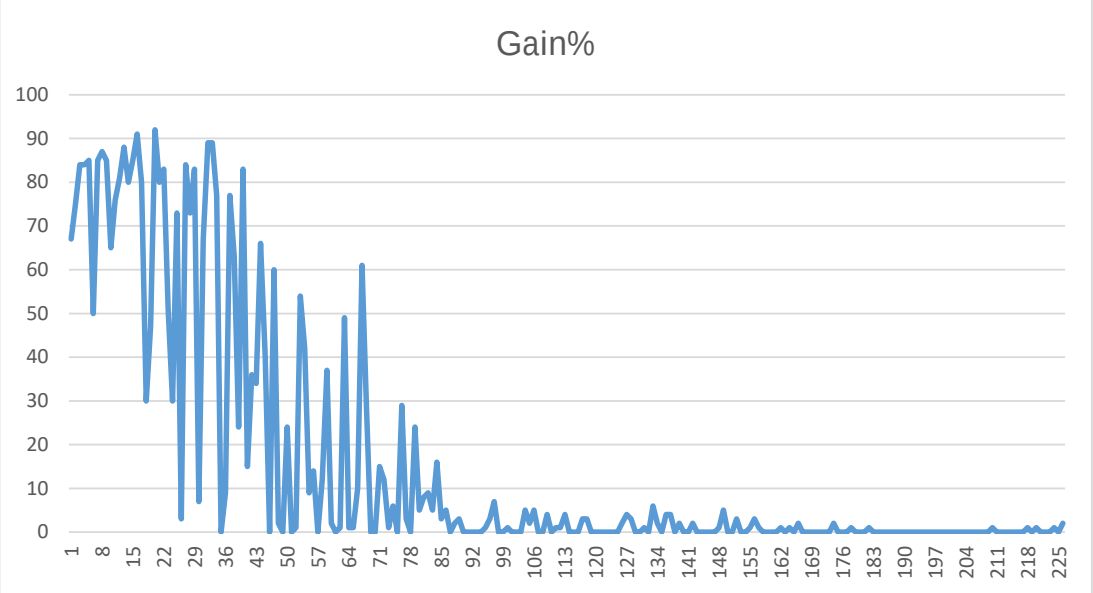


Rysunek 22. Wykres poprawy wykrywalności w funkcji numeru testu - kryterium min. zajętości pamięci

Na rysunkach 23 i 24 przedstawiono wyniki testów dla kryterium maksymalizacji szybkości działania. Widać, że metoda ICR zdecydowanie nadaje się do użycia dla tego kryterium. Można zauważyć, że miejscami metoda ICR powodowała nawet 90-punktowy wzrost wykrywalności przy bardzo niskich czasach generacji receptorów. Użytkownik, któremu zależy na szybkiej inicjacji algorytmu z pewnością doceni wysokie wzrosty wykrywalności.



Rysunek 23. Wykres czasu generacji receptorów w funkcji numeru testu - kryterium maks. szybkości działania



Rysunek 24. Wykres poprawy wykrywalności w funkcji numeru testu - kryterium maks. szybkości działania

4.2 Porównanie osiągów z innymi rozwiązaniami

Proponowany IDS został porównany ze znanymi z literatury rozwiązaniami. Należało w tym celu wybrać odpowiedni zbiór danych do wytrenowania i przetestowania systemu.

Szeroko wykorzystywanym w literaturze zbiorem danych jest zestaw pomiarów kwiatów irysa (ang. *Iris Flower Dataset*) [45, 47] udostępniony przez brytyjskiego statystyka i biologa Ronalda Fishera. Zbiór ten zawiera po 50 rekordów na każdy z trzech gatunków kwiatu irysa: *iris setosa*, *iris versicolor* i *iris virginica*. Każdy rekord zawiera 4 wartości pomiarów płatków kwiatów i informację o jego gatunku. Zbiór ten został wybrany w celu porównania osiągów IDS z innymi rozwiązaniami. W tabeli 48 przedstawiono przykładowe rekordy zbioru. Aby zapewnić kompatybilność z systemem IDS, zbiór ten został zapisany w formie ciągu 32-bitowych liczb typu zmiennoprzecinkowego.

Tabela 48. Przykładowe rekordy zbioru Iris Flower Dataset

#	**Długość działki kielicha**	**Szerokość działki kielicha**	**Długość płatka**	**Szerokość płatka**	**Gatunek**
1	5.1	3.5	1.4	0.2	I. setosa
2	4.9	3.0	1.4	0.2	I. setosa
3	4.7	3.2	1.3	0.2	I. setosa
4	4.6	3.1	1.5	0.2	I. setosa
5	5.0	3.6	1.4	0.3	I. setosa
...					

Algorytmami wybranymi do porównania osiągów były algorytmy klasyczne NSA-R (gdzie receptory dobierane są losowo) [26], NSA-T (w którym receptory

dobierane są przy pomocy wzorców) [77] oraz algorytmy MILA (ang. *multilevel immune learning algorithm*) [11], V-detector [42], FB-NSA [47], i zaproponowany w pracy algorytm ICR. Liczbę receptorów do generacji w przypadku algorytmów NSA-R i MILA ustalono na 1000, przy czym algorytm MILA ze względu na swoją konstrukcję zawierał 1000 receptorów typu T i 1000 grup receptorów typu B. Przyjęto, że parametr m (próg aktywacji) dla algorytmów wyniósł 6, a parametr l (długość receptora) był w przypadku metody ICR testowany w zakresie $\{16, 17, ..., 28\}$. Parametr l_I (długość receptorów międzykomórkowych) do testów ustalony został na wartość 16, a wartość m_I (próg aktywacji receptorów międzykomórkowych) testowano w zakresie $\{2, 3, ..., 8\}$. Testowanymi osiągami były parametry TP%, czyli średnia wykrywalność anomalii, FP%, czyli średnia liczba fałszywych alarmów i R_n, czyli średnia liczba receptorów w zbiorze $\mathbf{R}$ (w przypadku algorytmu ICR, sumy zbiorów $\mathbf{R}$ i $\mathbf{R_I}$).

Algorytm ICR trenowany był dla każdego gatunku kwiatów osobno. Algorytm trenowano najpierw przy pomocy wszystkich rekordów gatunku *iris setosa*, a następnie testowano korzystając z 100 losowych rekordów z dwóch pozostałych gatunków. Wyniki odnotowywano, a następnie algorytm trenowano tym razem przy pomocy tylko 50% rekordów gatunku *iris setosa*. 100 testów przeprowadzano ponownie, wyniki zapisywano, po czym całą procedurę powtarzano tym razem dla gatunku *iris versicolor*, i tak dalej, aż przetestowano wszystkie 3 gatunki. Takie same założenia zostały przyjęte dla pozostałych algorytmów.

Tabela 49 zawiera osiągi algorytmów dla poszczególnych gatunków kwiatów.

Tabela 49. Porównanie z innymi rozwiązaniami dla gatunku Iris setosa

Zbiór danych	**Algorytm**	**TP%**	**FP%**	$\mathbf{R_n}$
Iris setosa 100%	NSA-R	100	0	1000
	NSA-T	87.18	0	77.95
	MILA	95.16	0	1000
	V-detector	99.98	0	20
	FB-NSA	100	0	83
	ICR	94.84	0	111.3
Iris setosa 50%	NSA-R	100	11.18	1000
	NSA-T	97.44	6.1	91.46
	MILA	94.02	8.42	1000
	V-detector	99.97	1.32	16.44
	FB-NSA	100	2.76	76.97
	ICR	98.17	6.7	124.81

Dla pełnego zbioru trenującego *iris setosa*, można zaobserwować, że algorytmy NSA-R i FB-NSA osiągnęły 100% wykrywalności. Jednak w przypadku NSA-R wynik ten można było osiągnąć lokując aż 1000 receptorów w pamięci. Algorytmy ICR, MILA i V-detector osiągnęły wysoką wykrywalność na poziomie ponad 90%. Należy jednak zauważyć, że algorytm MILA wygenerował 1000 receptorów, ale wykazał mniejszą wykrywalność niż NSA-R, co czyni go najmniej opłacalnym dla tego zbioru danych w tej konfiguracji. Algorytm V-detector uzyskał imponującą wykrywalność na poziomie 99.98% przy 20 wygenerowanych receptorach. Metoda ICR pozwoliła na wykrycie anomalii w 94.84% przypadków przy średniej liczbie 111.3 wygenerowanych receptorów, co dla tego zbioru danych plasuje ją na 5. miejscu pod względem wykrywalności, a na miejscu 4. pod kątem jak najmniejszej liczby wygenerowanych receptorów. Wyniki są jednak porównywalne.

Warto zwrócić uwagę na fakt, że żadna z metod nie wykryła fałszywie żadnej anomalii. Ma to związek z całkowitym wytrenowaniem algorytmu negatywnej selekcji i jest to możliwe tylko w momencie posiadania pełnej wiedzy na temat zbioru fragmentów własnych.

Sytuacja zmieniła się, gdy dane trenujące zostały ograniczone do 50% danych zbioru *iris setosa*. Wykrywalności wszystkich metod z wyjątkiem MILA wzrosły, ale ze względu na ograniczone dane trenowania nie wszystkie rekordy zbioru zostały zaklasyfikowane jako fragmenty własne. Spowodowało to fałszywe wykrycia rekordów gatunku *iris setosa* jako gatunku obcego. Parametr FP% wyniósł 11.18 dla prostego algorytmu NSA-R, a najlepiej w tym teście wypadł algorytm V-detector, który osiągnął niski poziom 1.32 FP%. Algorytm ICR mimo osiągnięcia w tym teście 98.17% wykrywalności wykazał się dość dużą zajętością pamięci przez receptory w porównaniu do innych metod.

Tabela 50. Porównanie z innymi rozwiązaniami dla gatunku Iris versicolor

Zbiór danych	**Algorytm**	**TP%**	**FP%**	$\mathbf{R_n}$
Iris versicolor 100%	NSA-R	95.67	0	1000
	NSA-T	82.05	0	76.82
	MILA	84.37	0	1000
	V-detector	85.95	0	153.24
	FB-NSA	92	0	299
	ICR	88.27	0	110.24
Iris versicolor 50%	NSA-R	96	22.2	1000
	NSA-T	93.19	15.5	95.72
	MILA	84.46	19.6	1000
	V-detector	88.3	8.42	110.08
	FB-NSA	95.17	10.52	282.61
	ICR	97.9	16.1	129.14

W przypadku gdy zbiorem trenującym był pełny zbiór *iris versicolor* wykrywalności wszystkich algorytmów nieco spadły względem poprzedniego gatunku. Można zauważyć, że metodą która miała najmniejszy spadek wykrywalności jest metoda NSA-R (o 4.33 punktów procentowych). W porównaniu, wykrywalności algorytmów NSA-T, MILA, V-detector, FB-NSA i ICR obniżyły się odpowiednio o 5.13, 10.79, 14.03, 8 i 6.57 punktów procentowych. Algorytm ICR uzyskał wykrywalność lepszą od wykrywalności algorytmów NSA-T, MILA i V-detector, jednak mniej receptorów wygenerował tylko względem algorytmów MILA i V-detector. Najwyższą wykrywalność osiągnął algorytm FB-NSA, który uzyskał ten parametr na poziomie 92%. Jego wykonanie spowodowało jednak generację 299 receptorów. Oszczędniejszy był pod tym względem proponowany algorytm ICR, który utworzył średnio tylko 110.24 receptorów, uzyskując wykrywalność

niższą tylko o 3.73 punkty procentowe.

Dla zbioru trenującego będącego połową zbioru *iris versicolor*, można ponownie jak w przypadku poprzedniego gatunku zaobserwować zarówno zwiększenie liczby fałszywie wykrytych anomalii jak i zwiększenie wykrywalności. Algorytmem, który uzyskał najlepszą wykrywalność w tym teście jest proponowany algorytm ICR. Testy wykazały, że podejście to skutkowało wykryciem 97.9% anomalii, co jest wynikiem o prawie 2 punkty procentowe lepszym niż drugie miejsce w tym teście, czyli algorytm NSA-R. Poprawa ta jest dobra mając na uwadze, że średnia liczba receptorów wyniosła 129.14, a w przypadku NSA-R, 1000. Algorytmy NSA-T i V-detector osiągnęły niższe liczby receptorów, ale także niższą wykrywalność. Metoda MILA okazała się jedną z mniej skutecznych w tym teście, osiągając wykrywalność na poziomie 84.46% przy 1000 receptorach.

Tabela 51. Porównanie z innymi rozwiązaniami dla gatunku Iris virginica

Zbiór danych	**Algorytm**	**TP%**	**FP%**	$\mathbf{R_n}$
Iris virginica 100%	NSA-R	90.38	0	1000
	NSA-T	79.49	0	80.15
	MILA	75.75	0	1000
	V-detector	81.87	0	218.36
	FB-NSA	94	0	207
	ICR	94.1	0	115.32
Iris virginica 50%	NSA-R	97.18	33.26	1000
	NSA-T	90.91	25.39	97.72
	MILA	88.96	24.98	1000
	V-detector	93.58	13.18	108.12
	FB-NSA	95.78	16.64	194.37
	ICR	96.08	28	132.88

Dla pełnego zbioru *iris virginica* można spostrzec, że spośród badanych najlepszą wykrywalność osiągnął ponownie algorytm ICR. Receptory międzykomórkowe zastosowane w tym algorytmie pozwoliły zwiększyć wykrywalność względem metody NSA-T o 14.61 punktów procentowych, dając średnio 94.1 TP%. Warto zauważyć, że algorytm FB-NSA, który miał porównywalną wykrywalność na poziomie 94%, wykazał prawie dwukrotną zajętość pamięci przez receptory. W tym przypadku algorytm ICR wygenerował średnio 115.32 receptorów na test, a algorytm FB-NSA wygenerował ich 207. Pozostałe metody osiągały od 75.75 do 90.38 TP%. Jedyną metodą, która wygenerowała mniej receptorów niż ICR, była metoda NSA-T, która wypełniła zbiór receptorów średnio osiemdziesięcioma ich egzemplarzami.

W przypadku 50-procentowego zbioru *iris virginica* jako trenującego zbioru danych można zaobserwować wysokie parametry FP%, które na tym etapie sięgnęły już w przypadku niektórych algorytmów aż 33.26%. Algorytmem, który osiągnął ten pułap jest NSA-R. Brak połowy danych fragmentów własnych w tym zbiorze wykazał najmniejszy wpływ na algorytmy V-detector i FB-NSA, które odpowiednio uzyskały parametr FP% na poziomach 13.18 i 16.64. Wskaźnik TP% dla FB-NSA wyniósł 95.78% podczas gdy algorytm ICR był pod tym względem o 0.3 punkty procentowe lepszy. Metoda FB-NSA wykazała niższy od ICR wskaźnik fałszywych alarmów, jednak było to możliwe tylko dzięki znacznie większej liczbie receptorów w pamięci.

Można zauważyć, że sam zbiór danych ma wpływ na wykrywalności każdej z metod. Dalsze badania w temacie mogą uwzględniać weryfikację entropii zbioru danych według jednej z dostępnych metod [33, 83].

Podsumowując powyższe rozważania, można skonstatować, że w przypadku zbioru *iris setosa* metoda ICR osiągnęła wyniki porównywalne do wyników uzyskanych przez pozostałe badane algorytmy. Dla zbioru *iris versicolor* metoda

FB-NSA osiągnęła lepszą wykrywalność od metody ICR, ale metoda ICR miała lepszą opłacalność biorąc pod uwagę stosunek znacznie niższej zajętości pamięci przez receptory do wspomnianej wykrywalności. W przypadku algorytmów wytrenowanych przy pomocy pełnego zbioru *iris virginica* algorytm ICR osiągnął zarówno najlepszą wykrywalność jak i bardzo korzystną zajętość pamięci przez receptory.

Rozdział 5

Wnioski i podsumowanie

W pracy przedstawiono nowy algorytm poszukiwania infekcji w programach komputerowych. Zastosowano w nim autorską metodę receptorów międzykomórkowych. Tak opracowany algorytm został później przetestowany i porównany z osiągnięciami znanymi z literatury dla różnych parametrów.

Na podstawie testów można wywnioskować, że najistotniejszym dla osiągnięcia wysokiej wykrywalności jest parametr m, czyli próg aktywacji. Zbyt niska wartość tego parametru dla receptorów podstawowych lub pomocniczych niweluje wzrost wykrywalności. Okazało się, że proponowana metoda ma średnio lepsze wykrywalności na poziomie 17%. Ulepszony algorytm okazał się znakomitym wyborem, jeżeli wymagany jest szybki czas generacji receptorów lub możliwie najniższa zajętość pamięci przez receptory.

Badania eksperymentalne wykazały, że metoda receptorów międzykomórkowych osiągnęła porównywalne lub lepsze wyniki od algorytmów znanych z literatury, w tym znanych z literatury algorytmów wykorzystujących sztuczne systemy immunologiczne.

Głównymi osiągnięciami pracy są:

- analiza wad i zalet istniejących algorytmów sztucznych systemów immunologicznych,
- zaprojektowanie własnego systemu wykrywania intruzji (IDS) i jego implementacja,
- optymalizacja drzew binarnych w celu redukcji liczby receptorów w pamięci,
- opracowanie własnego algorytmu detekcji infekcji programów komputerowych, wprowadzającego receptory międzykomórkowe do algorytmów opartych o negatywną selekcję,
- badania eksperymentalne ulepszonej metody i jej weryfikacja za pomocą analizy porównawczej z innymi rozwiązaniami.

Uwzględniając osiągnięcia, cel pracy został osiągnięty, a teza dowiedziona.

Dalsze badania w temacie mogą uwzględniać na przykład zagadnienie naprawy zainfekowanych ciągów własnych i wpływ jakości implementacji systemu na osiągi.

Spis rysunków

Spis tablic

Literatura

[1] Abdolahnezhad, M. R. and Banirostam, T. [2016], 'Improved negative selection algorithm for email spam detection application', *Int. J. of Advanced Research in Electronics and Communication Engineering (IJARECE)* **5**(4), 956–960.

[2] Alrubayyi, H., Goteng, G., Jaber, M. and Kelly, J. [2021], 'A novel negative and positive selection algorithm to detect unknown malware in the iot', *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* pp. 1–6.

[3] Ayara, M., Timmis, J., de Lemos, R., De Castro, L. N. and Duncan, R. [2002], 'Negative selection: How to generate detectors', *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)* pp. 182–196.

[4] Balachandran, S., Dasgupta, D., Nino, F. and Garrett, D. [2007], 'A framework for evolving multi-shaped detectors in negative selection', *2007 IEEE Symposium on Foundations of Computational Intelligence* pp. 401–408.

[5] Balicki, J. [2006], 'Negative selection with ranking procedure in tabu-based multi-criterion evolutionary algorithm for task assignment', *Computational Science – ICCS 2006* pp. 863–870.

[6] Brown, J., Anwar, M. and Dozier, G. [2016], 'Detection of mobile malware: An artificial immunity approach', *2016 IEEE Security and Privacy Workshops (SPW)* pp. 74–80.

[7] Chen, W., Li, T., Liu, X. and Zhang, B. [2011], 'A negative selection algorithm based on hierarchical clustering of self set', *Science China Information Sciences* **56**(8), 1–13.

[8] Chikh, R. and Chikhi, S. [2017], 'Clustered negative selection algorithm and fruit fly optimization for email spam detection', *Journal of Ambient Intelligence and Humanized Computing* **10**(1), 143–152.

[9] Dasgupta, D. [1999], 'Immunity-based intrusion detection system: A general framework', *Proceedings of 22nd National Information Systems Security Conference* pp. 147–160.

[10] Dasgupta, D. and Forrest, S. [1999], *An Anomaly Detection Algorithm Inspired by the Immune System*, Springer-Verlag, p. 262–277.

[11] Dasgupta, D., Yu, S. and Majumdar, N. S. [2003], 'Mila — multilevel immune learning algorithm', *Genetic and Evolutionary Computation — GECCO 2003* pp. 183–194.

[12] Datta, A., Jha, S., Li, N., Melski, D. and Reps, T. [2010], 'Analysis techniques for information security', *Synthesis Lectures on Information Security, Privacy, and Trust* **2**(1), 1–164.

[13] De Boer, R. J. and Perelson, A. S. [1993], 'How diverse should the immune system be?', *Proceedings of the Royal Society of London. Series B: Biological Sciences* **252**(1335), 171–175.

[14] De Castro, L. N. and von Zuben, F. J. [2001], 'Immune and neural network models: theoretical and empirical comparisons', *International Journal of Computational Intelligence and Applications* **01**(03), 239–257.

[15] De Castro, L. and von Zuben, F. [2000], 'The clonal selection algorithm with engineering applications', *Genetic and Evolutionary Computation Conference, Workshop Proceedings of GECCO'00* pp. 36–37.

[16] De Castro, L. and Von Zuben, F. [2002], 'Learning and optimization using the clonal selection principle', *IEEE Transactions on Evolutionary Computation* **6**(3), 239–251.

[17] Delona, C., Haripriya, P. and Anju, J. [2017], 'Negative selection algorithm: A survey', *International Journal of Science, Engineering and Technology Research (IJSETR)* **6**(4), 711–715.

[18] Deng, H. and Yang, T. [2020], 'A negative selection algorithm based on adaptive immunoregulation', *2020 5th International Conference on Computational Intelligence and Applications (ICCIA)* pp. 177–182.

[19] D'haeseleer, P., Forrest, S. and Helman, P. [1996], 'An immunological approach to change detection: algorithms, analysis and implications', *Proceedings 1996 IEEE Symposium on Security and Privacy* .

[20] Dixon, S. E. [2010], Studies on Real-Valued Negative Selection Algorithms for Self-Nonself Discrimination, PhD thesis, California Polytechnic State University.

[21] Domingo-Ferrer, J., Muralidhar, K. and Bras-Amoros, M. [2020], 'General confidentiality and utility metrics for privacy-preserving data publishing based on the permutation model', *IEEE Transactions on Dependable and Secure Computing* pp. 2506–2517.

[22] Elahi, A. [2018], *Computer Systems: Digital Design, Fundamentals of Computer Architecture and Assembly Language*, 1 edn, Springer, Cham.

[23] Fadli, Z. M. and Jantan, A. [2011], 'An approach for malware behavior identification and classification', *2011 3rd International Conference on Computer Research and Development* .

[24] Fakhari, S. N. S. and Moghadam, A. M. E. [2011], 'Nssac: Negative selection-based self adaptive classifier', *2011 International Symposium on Innovations in Intelligent Systems and Applications* pp. 29–33.

[25] Farmer, J., Packard, N. H. and Perelson, A. S. [1986], 'The immune system, adaptation, and machine learning', *Physica D: Nonlinear Phenomena* **22**(1-3), 187–204.

[26] Forrest, S., Perelson, A., Allen, L. and Cherukuri, R. [1994], 'Self-nonself discrimination in a computer', *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy* pp. 202–212.

[27] Gao, X., Ovaska, S. and Wang, X. [2006], 'Genetic algorithms-based detector generation in negative selection algorithm', *2006 IEEE Mountain Workshop on Adaptive and Learning Systems* pp. 133–137.

[28] Gong, M., Zhang, J., Ma, J. and Jiao, L. [2012], 'An efficient negative selection algorithm with further training for anomaly detection', *Knowledge-Based Systems* **30**, 185–191.

[29] Gonzalez, F., Dasgupta, D. and Kozma, R. [2002], 'Combining negative selection and classification techniques for anomaly detection', *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)* pp. 705–710.

[30] González, F. and Dasgupta, D. [2003], 'Anomaly detection using real-valued negative selection', *Genetic Programming and Evolvable Machines* **4**(4), 383–403.

[31] González, F., Dasgupta, D. and Gómez, J. [2003], 'The effect of binary matching rules in negative selection', *Genetic and Evolutionary Computation — GECCO 2003* pp. 195–206.

[32] González, F., Dasgupta, D. and Niño, L. F. [2003], 'A randomized real-valued negative selection algorithm', *Lecture Notes in Computer Science* **2787**, 261–272.

[33] Guseva, A. I. and Kuznetsov, I. A. [2017], 'The use of entropy measure for higher quality machine learning algorithms in text data processing', *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)* .

[34] Haag, C. R., Lamont, G. B., Williams, P. D. and Peterson, G. L. [2007], 'An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions', *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation - GECCO '07* pp. 420–435.

[35] Hambouz, A., Shaheen, Y., Manna, A., Al-Fayoumi, M. and Tedmori, S. [2019], 'Achieving data integrity and confidentiality using image steganography and hashing techniques', *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* .

[36] Helman, P. and Forrest, S. [1994], *An efficient algorithm for generating random antibody strings*, Technical Report CS-94-07.

[37] Hofmeyr, S. [1999], An Immunological Model of Distributed Detection and Its Application to Computer Security, PhD thesis.

[38] Hofmeyr, S. A. [2000], *An interpretative introduction to the immune system*, Oxford University Press.

[39] Idris, I., Selamat, A. and Omatu, S. [2014], 'Hybrid email spam detection model with negative selection algorithm and differential evolution', *Engineering Applications of Artificial Intelligence* **28**, 97–110.

[40] Idris, I., Selamat, A., Thanh Nguyen, N., Omatu, S., Krejcar, O., Kuca, K. and Penhaker, M. [2015], 'A combined negative selection algorithm–particle swarm optimization for an email spam detection system', *Engineering Applications of Artificial Intelligence* **39**, 33–44.

[41] Ji, Z. [2006], Negative selection algorithms: From the thymus to V-detector, PhD thesis, The University of Memphis.

[42] Ji, Z. and Dasgupta, D. [2004], 'Real-valued negative selection algorithm with variable-sized detectors', *Genetic and Evolutionary Computation – GECCO 2004* pp. 287–298.

[43] Ji, Z. and Dasgupta, D. [2005], 'Estimating the detector coverage in a negative selection algorithm', *Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05* pp. 281–288.

[44] Ji, Z. and Dasgupta, D. [2006], 'Applicability issues of the real-valued negative selection algorithms', *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06* .

[45] Kamal, P. and Bhusry, M. [2016], 'Artificial bee colony optimization based

negative selection algorithms to classify iris plant dataset', *International Journal of Computer Applications* **133**(10), 40–43.

[46] Kang, S., Veeravalli, B., Mi Aung, K. M. and Jin, C. [2014], 'An efficient scheme to ensure data availability for a cloud service provider', *2014 IEEE International Conference on Big Data (Big Data)* pp. 15–20.

[47] Li, D., Liu, S. and Zhang, H. [2015], 'Negative selection algorithm with constant detectors for anomaly detection', *Applied Soft Computing* **36**, 618–632.

[48] Li, G., Li, T., Zeng, J. and Li, H. [2010], 'An outlier robust negative selection algorithm inspired by immune suppression', *Journal of Computers* **5**(9).

[49] Li, Z., Li, T., He, J., Zhu, Y. and Wang, Y. [2021], 'A hybrid real-valued negative selection algorithm with variable-sized detectors and the k-nearest neighbors algorithm', *Knowledge-Based Systems* **232**, 107477.

[50] Lu, T., Zhang, L., Wang, S. and Gong, Q. [2017], 'Ransomware detection based on v-detector negative selection algorithm', *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)* pp. 531–536.

[51] Marciniak, J., Wawryn, K. and Widulinski, P. [2018], 'An artificial immune negative selection algorithm to control water temperature in the outlet of the chamber', *2018 International Conference on Signals and Electronic Systems (ICSES)* pp. 236–241.

[52] Mixia, L., Dongmei, Y., Qiuyu, Z. and Honglei, Z. [2007], 'Network security risk assessment and situation analysis', *2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID)* .

[53] Mrozek, B. [2022], 'Algorytm selekcji klonalnej w zastosowaniu do strojenia regulatorów rozmytych', *XIV Konferencja Naukowo-Techniczna "Automatyzacja - Nowosci i Perspektywy, Automation 2010"*. .

[54] Nguyen, V., Le, T., Pham, T., Dinh, M. and Le, T. H. [2014], 'Kernel-based semi-supervised learning for novelty detection', *2014 International Joint Conference on Neural Networks (IJCNN)* pp. 4129–4136.

[55] Nisan, N. and Schocken, S. [2005], *The Elements of Computing Systems: Building a Modern Computer from First Principles*, 1 edn, The MIT Press.

[56] Nunes de Castro, L. and Von Zuben, F. J. [2002], 'ainet: An artificial immune network for data analysis', *Data Mining: A Heuristic Approach* pp. 231–260.

[57] Oprea, M. L. [1999], Antibody repertoires and pathogen recognition: the role of germline diversity and somatic hypermutation, PhD thesis, The University of New Mexico.

[58] Pamukov, M. E. and Poulkov, V. K. [2017], 'Multiple negative selection algorithm: Improving detection error rates in iot intrusion detection systems', *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* pp. 543–547.

[59] Percus, J. K., Percus, O. E. and Perelson, A. S. [1993], 'Predicting the size of the t-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination.', *Proceedings of the National Academy of Sciences* **90**(5), 1691–1695.

[60] Perelson, A. S. and Oster, G. F. [1979], 'Theoretical studies of clonal selection: Minimal antibody repertoire size and reliability of self-non-self discrimination', *Journal of Theoretical Biology* **81**(4), 645–670.

[61] Prathyusha, D. J. and Kannayaram, G. [2020], 'A cognitive mechanism for mitigating ddos attacks using the artificial immune system in a cloud environment', *Evolutionary Intelligence* **14**(2), 607–618.

[62] Reese, L. [1989], 'Challenges faced today by computer security practitioners', *[Proceedings] Fifth Annual Computer Security Applications Conference* .

[63] Reznik, L. [2022], *Intrusion Detection Systems*, 1 edn, Wiley-IEEE Press, pp. 109–176.

[64] Saleh, A. J., Karim, A., Shanmugam, B., Azam, S., Kannoorpatti, K., Jonkman, M. and Boer, F. D. [2019], 'An intelligent spam detection model based on artificial immune system', *Information* **10**(6), 209.

[65] Saurabh, P. and Verma, B. [2014], 'A novel immunity inspired approach for anomaly detection', *International Journal of Computer Applications* **94**(15), 14–19.

[66] Selahshoor, F., Jazayeriy, H. and Omranpour, H. [2019], 'Intrusion detection systems using real-valued negative selection algorithm with optimized detectors', *2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)* pp. 1–5.

[67] Shah, K. and Singh, D. K. [2015], 'A survey on data mining approaches for dynamic analysis of malwares', *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)* pp. 495–499.

[68] Somayaji, A., Hofmeyr, S. and Forrest, S. [1997], 'Principles of a computer immune system', *Proceedings of the 1997 workshop on New security paradigms - NSPW '97* pp. 75–82.

[69] Stibor, T., Timmis, J. and Eckert, C. [2005], 'A comparative study of real-valued negative selection to statistical anomaly detection techniques', *Lecture Notes in Computer Science* **3627**, 262–275.

[70] Tosin, S.-I. and Gbenga, J. [2020], 'Negative selection algorithm based intrusion detection model', *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)* pp. 202–206.

[71] Wawryn, K. and Widulinski, P. [2019], 'A human immunity inspired algorithm to detect infections in a computer program', *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"* pp. 381–385.

[72] Wawryn, K. and Widulinski, P. [2020], 'Detection of anomalies in compiled computer program files inspired by immune mechanisms using a template method', *Journal of Computer Virology and Hacking Techniques* **17**(1), 47–59.

[73] Wawryn, K. and Widulinski, P. [2022], 'Detekcja anomalii w plikach za pomoca wybranych algorytmów inspirowanych mechanizmami immunologicznymi', *Zeszyty Naukowe Wydzialu Elektroniki i Informatyki Politechniki Koszalinskiej* **14**, 23–41.

[74] Widulinski, P. [2019], 'Badanie wykrywalnosci anomalii w pliku monitorowanym przez system wykrywania intruzów w zaleznosci od parametrów generacji receptorów', *Zeszyty Naukowe Wydzialu Elektroniki i Informatyki Politechniki Koszalinskiej* **15**, 83–94.

[75] Widulinski, P. and Wawryn, K. [2020], 'A human immunity inspired intrusion detection system to search for infections in an operating system', *2020*

27th International Conference on Mixed Design of Integrated Circuits and System (MIXDES) pp. 187–191.

[76] Widulinski, P. and Wawryn, K. [2021], 'A study of detection probabilities and real-world testing of a human immunity inspired intrusion detection system', *2021 28th International Conference on Mixed Design of Integrated Circuits and System* pp. 261–264.

[77] Wierzchon, S. [2001], *Sztuczne systemy immunologiczne. Teoria i zastosowania*, 1 edn, Akademicka Oficyna Wydawnicza EXIT.

[78] Wierzchon, S. T. [2000], 'Generating optimal repertoire of antibody strings in an artificial immune system', *Intelligent Information Systems. Advances in Soft Computing* **4**, 119–133.

[79] Wierzchon, S. T. [2002], 'Function optimization by the immune metaphor', *Task Quarterly* **6**(3), 1–16.

[80] Wright, S. A. [2019], 'Privacy in iot blockchains: with big data comes big responsibility', *2019 IEEE International Conference on Big Data (Big Data)* pp. 5282–5291.

[81] Wu, B., Lu, T., Zheng, K., Zhang, D. and Lin, X. [2014], 'Smartphone malware detection model based on artificial immune system', *China Communications* **11**(13), 86–92.

[82] Yang, B. and Yang, M. [2020], 'Data-driven network layer security detection model and simulation for the internet of things based on an artificial immune system', *Neural Computing and Applications* **33**(2), 655–666.

[83] Yang, L., Xie, T., Yu, J., Zhai, L. and Tian, Q. [2020], 'Data processing based on variance weighted information entropy and maximally stable extremal

regions', *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)* .

[84] Yang, T., Chen, W. and Li, T. [2017], 'An antigen space density based real-value negative selection algorithm', *Applied Soft Computing* **61**, 860–874.

[85] Ying, T. [2016], *Artificial Immune System*, Wiley-IEEE Press, pp. 1–25.

[86] Zeng, J., Qin, R. and Tang, W. [2016], 'An extended negative selection algorithm for unknown malware detection', *Journal of Computational and Theoretical Nanoscience* **13**(6), 4010–4017.

[87] Zhang, F. and Ma, Y. [2016], 'Integrated negative selection algorithm and positive selection algorithm for malware detection', *2016 International Conference on Progress in Informatics and Computing (PIC)* pp. 605–609.

[88] Zhang, P., Wang, W. and Tan, Y. [2011], 'A malware detection model based on a negative selection algorithm with penalty factor', *SCIENTIA SINICA Informationis* **41**(7), 798–812.

[89] Zhao, G., Xuan, K., Rahayu, W., Taniar, D., Safar, M., Gavrilova, M. L. and Srinivasan, B. [2011], 'Voronoi-based continuous k nearest neighbor search in mobile navigation', *IEEE Transactions on Industrial Electronics* **58**(6), 2247–2257.

[90] Zhu, F., Chen, W., Yang, H., Li, T., Yang, T. and Zhang, F. [2017], 'A quick negative selection algorithm for one-class classification in big data era', *Mathematical Problems in Engineering* **2017**, 1–7.