PENERAPAN MONTE CARLO DAN LINEAR FUNCTION APPROXIMATION PADA SINGLE AGENT UNTUK LINGKUNGAN DAN LOKASI OBSTACLE YANG BERBEDA

SKRIPSI

Oleh :
M. AZIZ MAULANA AL QODAR R.
NIM. 18650010

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

PENERAPAN MONTE CARLO DAN LINEAR FUNCTION APPROXIMATION PADA SINGLE AGENT UNTUK LINGKUNGAN DAN LOKASI OBSTACLE YANG BERBEDA

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
M. Aziz Maulana Al Qodar R.
NIM. 18650010

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2023

HALAMAN PERSETUJUAN

PENERAPAN MONTE CARLO DAN LINEAR FUNCTION APPROXIMATION PADA SINGLE AGENT UNTUK LINGKUNGAN DAN LOKASI OBSTACLE YANG BERBEDA

SKRIPSI

Oleh :
M. Aziz Maulana
NIM. 18650010

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 1 Desember 2023

Pembimbing I	Pembimbing II.
Dr. Cresy Nugroho, M. T	Hani Nurhayati, M. T
NIP. 19710722 201101 1 001	NIP. 19780625 200801 2 006

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

iii

HALAMAN PENGESAHAN

PENERAPAN MONTE CARLO DAN LINEAR FUNCTION APPROXIMATION PADA SINGLE AGENT UNTUK LINGKUNGAN DAN LOKASI OBSTACLE YANG BERBEDA

SKRIPSI

Oleh :
M. Aziz Maulana
NIM. 18650010

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 12 Desember 2023

Susunan Dewan Penguji

Ketua Penguji	: Dr. M. Faisal, M. T NIP. 19740510 200501 1 007	( )
Anggota Penguji I	: Ahmad Fahmi Karami, M.Kom NIP. 198709092020121001	( )
Anggota Penguji II	: Dr. Fresy Nugroho, M. T NIP. 19710722 201101 1 001	( )
Anggota Penguji III	: Hani Nurhayati, M. T NIP. 19780625 200801 2 006	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

iv

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: M. Aziz Maulana
NIM	: 1865000
Fakultas / Prodi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Penerapan Monte Carlo dan Linear Function Approximation pada Single Agent untuk Lingkungan dan Lokasi Obstacle yang Berbeda.

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 14 Desember 2023
Yang membuat pernyataan,

M. Aziz Maulana
NIM.18650010

HALAMAN MOTTO

… Connecting ideas, building the future …

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur, penulis ingin menyampaikan penghargaan dan terima kasih kepada semua pihak yang telah memberikan dukungan, bimbingan, dan inspirasi dalam menyelesaikan penelitian ini. Terima kasih kepada Dr. Fresy Nugroho, M. T dan Hani Nurhayati, M. T yang telah memberikan arahan, motivasi, dan bimbingan yang berharga.

Ucapan terima kasih juga disampaikan kepada Dr. M Faisal M. T, Ahmad Fahmi Karami, M. Kom, Dr. Fresy Nugroho, M. T dan Hani Nurhayati, M. T sebagai dosen penguji yang telah memberikan atas masukan dan saran yang memperkaya hasil penelitian ini, Sehingga penelitian ini dapat terselesaikan.

Penghargaan dan terima kasih tak terhingga diberikan kepada keluarga, teman-teman, dan semua yang telah memberikan dukungan moril dan doa selama perjalanan penulisan skripsi ini. Semua kontribusi mereka telah menjadi pilar penting dalam kesuksesan penelitian ini. Terima kasih atas segala dukungan dan motivasi yang telah diberikan. Semoga hasil penelitian ini dapat bermanfaat dan memberikan kontribusi positif bagi perkembangan ilmu pengetahuan. Penulis berharap dapat terus berkembang dan berkontribusi lebih banyak lagi di masa depan

Akhir kata, penulis ingin menyampaikan rasa terima kasih kepada semua pihak yang tidak dapat disebutkan satu per satu namun telah memberikan dukungan dan motivasi dalam berbagai bentuk selama penulisan skripsi ini. Semoga segala

bantuan dan doa yang telah diberikan oleh keluarga tercinta dapat diberikan balasan yang setimpal oleh Allah SWT.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, hidayah, serta karunia-Nya yang telah melimpah dalam perjalanan penulisan skripsi ini. Skripsi dengan judul " Penerapan Monte Carlo dan Linear Function Approximation pada Single Agent untuk Lingkungan dan Lokasi Obstacle yang Berbeda " merupakan bagian dari upaya penulis dalam menyelesaikan pendidikan program studi Teknik Informatika di Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Tanpa dukungan doa, izin, dan dorongan dari berbagai pihak serta tekad penulis sendiri, Skripsi ini tidak akan dapat terealisasi. Oleh karena itu, penulis ingin menyampaikan apresiasi yang sangat besar kepada:

1. Prof. Dr. M. Zainuddin, M.A., selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Prof. Dr. Hj. Sri Hariani, M.Si., selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrul Kurniawan M.MT., IPM selaku Ketua Prodi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Dr. Fresy Nugroho, M. T dan Hani Nurhayati, M. T. selaku pembimbing skripsi. Terima kasih atas bimbingan, arahan, serta masukan yang berharga dari keduanya dalam menyelesaikan skripsi ini.
5. Dr. M Faisal, M. T dan Ahmad Fahmi Karami, M. Kom. selaku penguji skripsi. Terima kasih telah meluangkan waktu dan tenaga untuk membaca, menelaah, serta memberikan masukan dan evaluasi terhadap skripsi ini.

6. Keluarga tercinta, terutama orangtua dan saudara-saudari, atas dukungan, cinta, dan doa yang tak henti-hentinya diberikan. Ridho, semangat, serta dorongan yang berasal dari keluarga adalah sumber inspirasi utama penulis dalam menyelesaikan skripsi ini.

Terakhir, penulis ingin menyampaikan rasa terima kasih kepada semua pihak yang tidak dapat disebutkan satu per satu namun telah memberikan dukungan dan motivasi dalam berbagai bentuk selama penulisan skripsi ini.

Semoga skripsi ini dapat memberikan manfaat dan kontribusi yang positif bagi perkembangan ilmu pengetahuan di bidang yang relevan. Penulis menyadari bahwa skripsi ini tidak luput dari kekurangan dan keterbatasan. Oleh karena itu, penulis sangat mengharapkan kritik, saran, dan masukan yang membangun untuk pengembangan penelitian ini di masa yang akan datang.

Akhir kata, semoga Allah SWT senantiasa memberikan rahmat, hidayah, dan keberkahan-Nya kepada kita semua. Amin.

Malang, 12 Desember 2023

M. Aziz Maulana

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Maulana, Muhammad Aziz. 2023. Penerapan Monte Carlo Dan Linear Function Approximation Pada Single Agent Untuk Lingkungan Dan Lokasi Obstacle Yang Berbeda. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing:(I) Dr. Fressy Nugoho, M. T(II) Hani Nurhayati, M. T.

.

Kata kunci: monte carlo method, linear function approximation, single agent.

Penelitian ini bertujuan untuk melatih single agent dari algoritma monte carlo method dan linear function approximation, serta menguji dari single agent tersebut ke dalam 3 lingkungan yang telah dibuat. Monte Carlo dan Linear Function Approximation ini melibatkan pengajaran pada *agent* untuk dapat mempelajari, memahami, dan memutuskan tindakan apa yang harus diambil dalam suatu lingkungan untuk memaksimalkan akumulasi dari hadiah yang didapatkan oleh agent tersebut. Aspek penting di *Monte Carlo* dan *Linear Function Approximation* memiliki *agent* yang dapat mempelajari perilaku dengan baik di lingkungan, sehingga *agent* tidak memerlukan pengetahuan atau kontrol penuh atas lingkungan, namun hanya perlu untuk mampu berinteraksi serta mengumpulkan informasi di lingkungan. Dengan hasil yang didapatkan dari perbandingan uji coba ini yaitu monte carlo method lebih efektif dalam sebuah pengajaran pada agent.

ABSTRACT

Kurniawan, Muhammad Aziz. 2023. Penerapan Monte Carlo Dan Linear Function Approximation Pada Single Agent Untuk Lingkungan Dan Lokasi Obstacle Yang Berbeda. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing:(I) Dr. Fressy Nugoho, M. T(II) Hani Nurhayati, M. T.

.

This research aims to train a single agent using the Monte Carlo algorithm method and linear function approach, as well as testing the single agent in the 3 environments that have been created. Monte Carlo and Linear Function Approximation involve teaching agents to be able to learn, understand, and decide what actions to take in an environment to maximize the accumulation of rewards obtained by the agent. An important aspect in Monte Carlo and Linear Function Approximation is having an agent that can learn behavior well in the environment, so that the agent does not need full knowledge or control over the environment, but only needs to be able to interact and collect information in the environment. With the results obtained from this trial comparison, the Monte Carlo method is more effective in teaching agents.

Keywords : monte carlo method, linear function approximation, single agent.

الملخص

مولانا، محمد عزيز. 2023. تطبيق مونت كارلو وتقريب الوظيفة الخطية على عامل واحد لبيئات ومواقع العوائق المختلفة. البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: د. فريسي نوغوهو، الماجستير. المشرف الثاني: هاني نورحياتي، الماجستيرة.

الكلمات الرئيسية: طريقة مونت كارلو، تقريب الوظيفة الخطية، عامل واحد.

يهدف هذا البحث إلى تدريب عامل واحد من خوارزمية مونت كارلو وتقريب الوظيفة الخطية، واختبار العامل الواحد في 3 بيئات تم إنشاؤها. مونت كارلو وتقريب الوظيفة الخطية ينطوي على تعليم العملاء ليكونوا قادرين على التعلم والفهم وتحديد الإجراءات التي يجب اتخاذها في بيئة لزيادة تراكم المكافآت التي حصل عليها العامل. أحد الجوانب المهمة في مونت كارلو وتقريب الوظيفة الخطية هو أن العملاء يمكنهم تعلم السلوك جيدا في البيئة، لذلك لا يحتاج العملاء إلى معرفة كاملة أو تحكم في البيئة، ولكنهم يحتاجون فقط إلى أن يكونوا قادرين على التفاعل وجمع المعلومات في البيئة. مع النتائج التي تم الحصول عليها من مقارنة هذه التجربة، فإن طريقة مونت كارلو أكثر فعالية في تعليم العملاء.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era modern ini, kemajuan teknologi dan kompleksitas masalah yang dihadapi oleh berbagai disiplin ilmu menuntut pendekatan analitis yang lebih canggih. Salah satu cabang analisis numerik yang telah mendapat perhatian luas adalah metode Monte Carlo, yang mengandalkan simulasi acak untuk mengestimasi solusi dari permasalahan matematis yang kompleks.

Metode Monte Carlo telah menjadi pendekatan yang sangat populer untuk menangani masalah yang melibatkan ketidakpastian dan kompleksitas yang tinggi. Metode ini melibatkan penggunaan sampel acak untuk menghasilkan perkiraan atau estimasi numerik, sehingga sangat berguna dalam situasi di mana analisis matematis konvensional sulit diterapkan.

Sementara itu, linear function approximation adalah teknik lain yang mendalam dan efisien untuk mewakili hubungan antara variabel-variabel dalam suatu sistem. Penggunaan fungsi linear untuk mendekati fungsi yang kompleks memungkinkan penghematan komputasi dan mempermudah interpretasi model.

Ketika kita menghadapi masalah analisis numerik yang melibatkan fungsi-fungsi kompleks, penggunaan metode Monte Carlo seringkali dihadapkan pada kendala waktu komputasi yang tinggi. Di sinilah potensi untuk penggunaan metode Monte Carlo dengan linear function approximation muncul sebagai solusi yang menarik. Linear function approximation memungkinkan representasi aproksimatif

dari fungsi kompleks menggunakan fungsi linier, membuka peluang untuk meningkatkan efisiensi perhitungan yang dilakukan oleh metode Monte Carlo.

Banyak sekali penelitian yang menggunakan monte carlo, salah satunya penelitian yang dilakukan oleh [1] dimana penggunaan monte carlo untuk melakukan simulasi penjualan dapat membantu pihak toko dalam melakukan proses pengolahan data dan perkiraan jumlah penjualan barang pada periode berikutnya. Penelitian linear function approximation yang dilakukan oleh [2] dimana di dalam penelitiannya menggunakan linear function approximation untuk mengetahui kemampuan mahasiswa dalam membaca AlQuran, penguasaan ilmu tajwid, dan hubungan kedua variabel tersebut.

Monte Carlo dan Linear Function Approximation ini melibatkan pengajaran pada agent untuk dapat mempelajari, memahami, dan memutuskan tindakan apa yang harus diambil dalam suatu lingkungan untuk memaksimalkan akumulasi dari hadiah yang didapatkan oleh agent tersebut. Aspek penting di Monte Carlo dan Linear Function Approximation memiliki agent yang dapat mempelajari perilaku dengan baik di lingkungan, sehingga agent tidak memerlukan pengetahuan atau kontrol penuh atas lingkungan, namun hanya perlu untuk mampu berinteraksi serta mengumpulkan informasi di lingkungan[3]. Agent merupakan entitas yang mempelajari cara untuk berinteraksi dengan lingkungan untuk mencapai tujuan tertentu melalui pembelajaran secara mandiri. Pada pembelajaran agent diberikan lingkungan (environtment) yang diketahui maupun tidak diketahui untuk tempat dia bisa bereksplorasi.

Agent tidak akan diberitahu aksi mana yang harus diambil, melainkan agen harus menemukan aksi yang menghasilkan imbalan paling banyak dengan mencoba berinteraksi dan menemukan jalur untuk dapat mencapai suatu tujuan dengan menghindari obstacle yang diberikan. Allah berfirman pada QS. An-najm ayat 39-42 sebagai berikut.

وَأَنْ لَيْسَ لِلْإِنْسَانِ إِلَّا مَا سَعَى . وَأَنَّ سَعْيَهُ سَوْفَ يُرَى . ثُمَّ يُجْزَاهُ الْجَزَاءَ الْأَوْفَى . وَأَنَّ إِلَى رَبِّكَ الْمُنْتَهَى

"Bahwa manusia hanya memperoleh apa yang telah diusahakannya, bahwa sesungguhnya usahanya itu kelak akan diperlihatkan (kepadanya), kemudian dia akan diberi balasan atas (amalnya) itu dengan balasan yang paling sempurna, bahwa sesungguhnya kepada Tuhanmulah kesudahan (segala sesuatu*),"(Q.S an Najm: 39-42)*

Melalui ayat ini, Allah SWT berjanji memberi jawaban tepat kepada orang-orang yang mau berusaha keras. Setiap berjuang dan berikhtiar untuk memenuhi kebutuhan hidup, hendaknya diawali dengan niat karena Allah SWT. Sehingga, Allah SWT akan mengaruniakan pahala yang berkali-kali lipat. Kemudian, pahala tersebut akan menjadi bekal meraih kebahagiaan di akhirat. Tentu saja, surga merupakan jawaban tepat dari Allah SWT bagi hamba-hambaNya yang saleh.

Untuk meraih nirwana, seorang hamba perlu berikhtiar sekuat tenaga. Di antaranya melakukan perintah Allah SWT dan menjauhi larangan-Nya. Tidak hanya itu, shalat, zakat, puasa dan ibadah lainnya juga merupakan sarana meraih surga. Ibadah-ibadah tersebut harus dikerjakan dengan penuh sungguh-sungguh.

Penelitian terhadap agent yang dilakukan oleh [4]yang melakukan pengeditan memori dua arah untuk menghasilkan berbagai sub tujuan untuk

meningkatkan efiensi pembelajaran pada agent. Penelitian yang dilakukan oleh [5] yang menggunakan agent AI dalam bermain mode deathmatch pada game FPS yang hanya menggunakan piksel pada layar. Agent tersebut digunakan untuk 2 masalah, yang pertama untuk navigasi (menjelajahi peta serta mengumpulkan item), yang kedua ada aksi yang digunakan untuk melawan musuh ketika mereka terlihat.

Berdasarkan uraian diatas, maka penelitian ini akan mengajukan penerapan dari Monte Carlo dan Linear Function Approximation untuk melakukan perbandingan pada uji coba terhadap single agent di berbagai lingkungan yang dibuat serta lokasi obstacle yang berbeda

1.2 Rumusan Masalah

Bagaimana menerapkan *single agent monte carlo* dan *linear function approximation* untuk menghindari *obstacle* dan mencapai tujuan.

1.3 Tujuan Penelitian

Penelitian ini memiliki tujuan , antara lain :

a. Membuat *environment, agent,* dan *obstacle.*
b. Melakukan percobaan terhadap agen di berbagai lingkungan dengan menggunakan *Monte Carlo* dan *Linear Function Approximation* diterapkan pada *single agent*
c. Melakukan perbandingan pada percobaan yang diambil oleh *Monte Carlo* dan *Linear Function Approximation* diterapkan pada *single agent* di berbagai lingkungan yang dibuat.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini antara lain :

a. Penelitian ini dapat memahami konsep, metode dan algoritma yang mendasari *deep reinforcement learning* sehingga agent dapat memahami tentang sebuah kebijakan (*policy*), fungsi nilai, algoritma pelatihan, dan juga belajar untuk berinteraksi dengan lingkungan.
b. Penelitian pada *monte carlo* dan *linear function approximation* ini dapat digunakan sebagai alat Pendidikan dan landasan untuk penelitian lebih lanjut. Peneliti dapat memodifikasi terkait dengan parameter, kinerja, danknologi medis yang lebih maju berdasarkan hasil penelitian ini.

1.5 Batasan Penelitian

Penelitian ini dibatasi oleh hal berikut :

a. Penelitian ini akan menggunakan metode *Monte Carlo* dan *Linear Function Approximation* diterapkan pada *single agent.*
b. Penerapan lingkungan ada 3 yang dibuat yaitu, persegi, lingkaran, dan huruf.
c. Penentuan koordinat dari agent, obstacle, serta target.

BAB II
STUDI PUSTAKA

2.1 Penelitian Terkait

Pada Penelitian yang dilakukan oleh [6] yang menggunakan *monte carlo* untuk memprediksi data pemesan sehingga sistem barang dari pengadaan dapat optimal dan berguna bagi supplier Qshop. Dimana algoritma *monte carlo* tersebut digunakan untuk perhitungan barang dari pemesan konsumen dalam memprediksi 2 tahun kedepan, dengan mensimulasikan permintaan barang-barang aksesoris mobil 2 tahun dan untuk mendapatkan hasil prediksi pengadaan 2 tahun mendatang diQshop Batam.

Penelitian yang dilakukan oleh [7] yang menggunakan *monte carlo* untuk melakukan simulasi dalam memprediksi jumlah penumpang angkutan massa bus rapid. Dengan hasil yang diperoleh pada simulasi prediksi jumlah penumpang bus rapit transit untuk tahun 2018 memiliki rata-rata akurasi 82,43 % dengan begitu disimpulkan bahwa metode monte carlo bisa digunakan untuk membantu pihak pengelola bus rapit transit dalam mengetahui jumlah penumpang pada tahun yang akan datang.

Penelitian yang dilakukan oleh [8] yang menggunakan *monte carlo method* untuk perencanaan jalur pada robot. Penelitian tersebut menghasilkan pada environment 3, training 1, 2 dan 3, agent tidak berhasil menemukan rute ke tujuan dikarenakan dengan nilai learning rate yang sama yaitu $\alpha = 0.1$, pelatihan dilakukan secara lambat (pembaruan Q-value lambat) sehingga Qtable belum konvergen dalam 1000 episode training.

Penelitian yang dilakukan oleh [9] yang mengkombinasikan antara *Q learning* dengan *linear function approximation*. Pada penelitiannya itu menganalisis tentang konvergensi pada *Q learning* dengan *linear function approximation* dengan proses pengambilan dari keputusan markov yang dapat diobservasi supaya bisa teratasi.

Pada penelitian [10] yang membahas tentang *linear function approximation* apakah terbukti lebih efisien dalam penggunannya pada *reinforcement learning*. Pada penelitian tersebut menjelaskan bisa menyajikan untuk algoritma yang terbukti efisien dalam kondisi *runtime* maupun dalam penggunaan kompleksitas pada sample.

2.2 Agent

Agent adalah entitas yang cerdas dan adaptif yang belajar dari pengalaman, beradaptasi dengan perubahan lingkungan dan bertindak dalam suatu lingkungan untuk tujuan yang telah ditetapkan. *Agent* dapat menjadi orang, robot, anjing, cacing, angin, gravitasi, lampu, atau program komputer[11].

Agent purposive memiliki preferensi. Mereka lebih memilih beberapa negara bagian dunia daripada negara bagian lain, dan mereka bertindak untuk mencoba mencapai negara bagian yang paling mereka sukai. *Agent* non-tujuan dikelompokkan bersama dan disebut alam. Apakah suatu *agent* bertujuan atau tidak adalah asumsi model yang mungkin, atau mungkin tidak sesuai. Misalnya, untuk beberapa aplikasi mungkin tepat untuk memodelkan anjing sebagai purposive, dan untuk yang lain mungkin cukup memodelkan anjing sebagai non-purposive.

Jika seorang *agent* tidak memiliki preferensi, menurut definisinya ia tidak peduli di negara dunia mana ia berakhir, dan karenanya tidak peduli apa yang dilakukannya. Satu-satunya alasan untuk merancang sebuah *agent* adalah untuk menanamkannya dengan preferensi – untuk membuatnya lebih menyukai beberapa negara dunia dan mencoba untuk mencapainya[11]. Seorang *agent* tidak harus mengetahui preferensinya. Misalnya, termostat adalah agen yang merasakan dunia dan menyalakan atau mematikan pemanas. Ada preferensi yang disematkan di termostat, seperti menjaga penghuni ruangan pada suhu yang menyenangkan, meskipun termostat tidak tahu ini adalah preferensinya. Preferensi *agent* seringkali merupakan preferensi perancang *agent*, tetapi terkadang *agent* dapat diberi tujuan dan preferensi pada waktu berjalan.

Pada awalnya *agent* tidak mengetahui apa pengaruh dari tindakan terhadap keadaan lingkungan atau imbalan langsung apa yang diberikan yang dihasilkan dari tindakan *agent*, namun *agent* akan mencoba berbagai tindakan di berbagai lingkungan dan secara bertahap mempelajari lingkungan mana yang terbaik di tiap bagian untuk memaksimalkan imbalan dalam waktu jangka Panjang.

Seorang *agent* akan bertindak cerdas Ketika

a. apa yang dilakukan agen sesuai dengan keadaan dan tujuannya,
b. *agent* fleksibel terhadap lingkungan yang berubah dan tujuan yang berubah,
c. *agent* belajar dari pengalaman, dan
d. *agent* membuat pilihan yang tepat mengingat keterbatasan persepsi dan komputasinya. Seorang *agent* biasanya tidak dapat mengamati keadaan

dunia secara langsung; ia hanya memiliki ingatan yang terbatas dan ia tidak memiliki waktu tak terbatas untuk bertindak.

Agent komputasi adalah *agent* yang keputusannya tentang tindakannya dapat dijelaskan dalam istilah komputasi. Artinya, keputusan dapat dipecah menjadi operasi primitif yang dapat diimplementasikan dalam perangkat fisik. *Agent* komputasi bersifat relatif mudah untuk dibuat dibandingkan dengan model komputasi lainnya, hal ini terjadi karena inti dari code nya merupakan perilaku dari *agent*[11].

2.2.1 Agen yang Berada di Lingkungan

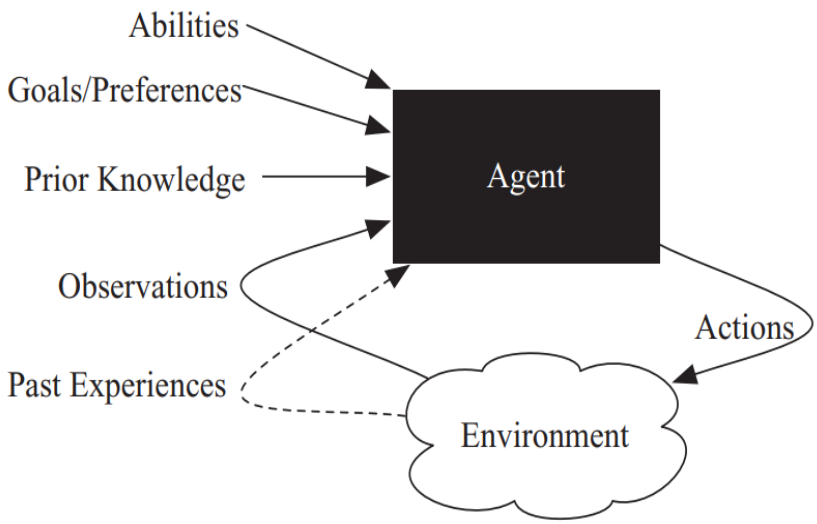
Agent dapat berupa, misalnya, kopling mesin komputasi dengan sensor fisik dan aktuator, yang disebut robot, di mana lingkungannya adalah pengaturan fisik. Agen bisa menjadi program yang bertindak dalam lingkungan komputasi murni – agen perangkat lunak [12].

Gambar 2.1 menunjukkan input dan output dari *agent*. Setiap saat, apa yang dilakukan agen bergantung pada *agent* tersebut

a. pengetahuan awal tentang *agent* dan lingkungan,
b. agent berinteraksi dengan lingkungan, yang terdiri dari :
c. pengamatan lingkungan saat ini dan
d. pengalaman masa lalu dari tindakan dan pengamatan sebelumnya, atau data lain, dari mana agen dapat belajar.
e. tujuan yang harus diusahakan untuk dicapai atau preferensi atas negara-negara dunia, Dan
f. kemampuan, yang merupakan tindakan 9rimitive yang mampu dilakukannya.

Lingkungan itu sendiri terdiri dari beberapa macam, namun salah satunya adalah lingkungan yang diketahui dan lingkungan yang tidak diketahui. Lingkungan yang diketahui merupakan sebuah lingkunga dimana *agent* memiliki

pemahaman yang lengkap tentang keadaan dan aturan yang mengaturnya. Sebagai contoh dalam permainan kartu solitaire, agen dapat mengetahui aturannya, namun *agent* tidak dapat melihat kartunya sebelum kartu tersebut dikeluarkan. Lingkungan yang tidak diketahui seperti dalam sebuah permainan video game, pada layar akan menampilkan seluruh keadaan dalam permainan, namun *agent* masih belum untuk mengetahui fungsi dari tombol-tombol yang ada sehingga harus mencoba terlebih dahulu[13].

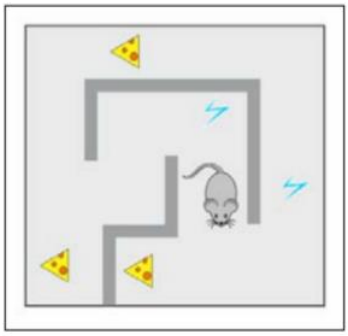


Gambar 2.1 Input output dari agent

Sistem agen terdiri dari agen dan lingkungannya. *Agent* menerima rangsangan dari lingkungan dan melakukan tindakan di lingkungan.

Agent terdiri dari badan dan pengontrol. Pengontrol menerima persepsi dari tubuh dan mengirimkan perintah ke tubuh. Tubuh mencakup sensor yang mengubah

rangsangan menjadi persepsi dan aktuator yang mengubah perintah menjadi tindakan. Seperti contoh yang ada di dalam buku yang di tulis oleh [12]



Gambar 2.2 Robot tikus pada lingkungan labirin

Pada lingkungan robot tersebut adalah sebuah labirin yang telah ditempati oleh robot beserta dengan makanan serta tantangan tersebut. Robot tersebut dapat mengambil tindakan seperti bergerak ke kiri atau kanan serta maju ke depan, dan robot tersebut setiap saat dapat untuk mengamati kondisi labirin secara keseluruhan untuk mengambil keputusan tentang tindakan yang diambil. Makanan serta listrik yang diberikan pada labirin tersebut merupakan sebuah sinyal hadiah diberikan kepada robot oleh labirin sebagai umpan balik terhadap tindakan yang telah dilakukan oleh robot tersebut.

Dua strategi luas telah dikejar dalam membangun *agent* di dalam lingkungan:

a. Yang pertama adalah menyederhanakan lingkungan dan membangun sistem penalaran kompleks untuk lingkungan sederhana ini. Misalnya, robot pabrik dapat melakukan tugas-tugas canggih di lingkungan pabrik yang direkayasa, tetapi mungkin tidak ada harapan di lingkungan alami. Sebagian besar kompleksitas masalah dapat dikurangi dengan menyederhanakan

lingkungan. Ini juga penting untuk membangun sistem praktis karena banyak lingkungan dapat direkayasa untuk membuatnya lebih sederhana bagi agen.

b. Strategi kedua adalah membangun *agent* sederhana di lingkungan alami. Hal ini terinspirasi dari melihat bagaimana serangga dapat bertahan hidup di lingkungan yang kompleks meskipun kemampuan nalar mereka sangat terbatas. Peneliti kemudian membuat *agent* memiliki lebih banyak kemampuan penalaran karena tugas mereka menjadi lebih rumit.

Salah satu keuntungan dari penyederhanaan lingkungan adalah memungkinkan kita membuktikan properti *agent* atau mengoptimalkan *agent* untuk situasi tertentu. Membuktikan properti atau pengoptimalan biasanya memerlukan model *agent* dan lingkungannya. *Agent* mungkin melakukan sedikit atau banyak penalaran, tetapi seorang pengamat atau perancang *agent* mungkin dapat memberi alasan tentang *agent* dan lingkungan. Misalnya, perancang mungkin dapat membuktikan apakah *agent* dapat mencapai suatu tujuan, apakah dapat menghindari masuk ke dalam situasi yang mungkin buruk bagi *agent* (sasaran keselamatan), apakah akan terjebak di suatu tempat (kehidupan), atau apakah itu pada akhirnya akan menyiasati setiap hal yang seharusnya dilakukan (keadilan).

2.2.2 Jumlah *Agent*

Jumlah *agent* pada DRL merujuk pada seberapa banyak entitas yang terlibat pada sebuah lingkungan untuk mempelajari suatu kebijakan (*policy*) yang optimal

untuk mengambil tindakan yang maksimal dalam interaksi dengan lingkungan tersebut mengambil sudut pandang *single agent*, dimensi jumlah *agent* mempertimbangkan apakah *agent* tersebut melakukannya[13]

a. penalaran *single agent*, hanya ada satu entitas atau *agent* yang beroperasi dalam lingkungan, di mana *agent* menganggap bahwa agen lain hanyalah bagian dari lingkungan. Ini adalah asumsi yang masuk akal jika tidak ada agen lain atau jika *agent* lain tidak akan mengubah apa yang mereka lakukan berdasarkan tindakan agen tersebut.
b. penalaran *multi agent*, ada lebih dari satu *agent* yang berinteraksi dengan lingkungan yang sama. di mana agen mempertimbangkan penalaran *agent* lain. Ini terjadi ketika ada *agent* cerdas lain yang tujuan atau preferensinya bergantung sebagian, pada apa yang dilakukan agen tersebut atau jika *agent* tersebut harus berkomunikasi dengan agen lain.

Penalaran di hadapan agen lain jauh lebih sulit jika *agent* dapat bertindak secara bersamaan atau jika lingkungan hanya dapat diamati sebagian.

2.3 Pembelajaran

Seleksi Fitur Dalam beberapa kasus, perancang *agent* mungkin memiliki model *agent* dan lingkungannya yang baik. Seringkali seorang desainer tidak memiliki model yang baik, dan seorang *agent* harus menggunakan data dari pengalaman masa lalunya dan sumber lain untuk membantunya memutuskan apa yang harus dilakukan.

Dimensi pembelajaran menentukan apakah

a. pengetahuan diberikan atau

b. pengetahuan dipelajari (dari data atau pengalaman masa lalu).

Belajar biasanya berarti menemukan model terbaik yang sesuai dengan data. Pembelajaran itu sendiri adalah bidang yang sangat besar tetapi tidak berdiri sendiri dari AI lainnya. Ada banyak masalah di luar pemasangan data, termasuk bagaimana menggabungkan pengetahuan latar belakang, data apa yang dikumpulkan, bagaimana merepresentasikan data dan representasi yang dihasilkan, bias pembelajaran apa yang sesuai, dan bagaimana pengetahuan yang dipelajari dapat digunakan untuk mempengaruhi bagaimana agen bertindak[13].

2.4 Limit Komputasi

Terkadang seorang agen dapat memutuskan tindakan terbaiknya dengan cukup cepat untuk dapat bertindak. Seringkali ada batasan sumber daya komputasi yang mencegah agen melakukan tindakan terbaik. Artinya, *agent* mungkin tidak dapat menemukan tindakan terbaik dengan cukup cepat dalam keterbatasan ingatannya untuk bertindak sementara tindakan itu masih merupakan hal terbaik untuk dilakukan. Misalnya, mungkin tidak banyak gunanya mengambil 10 menit untuk mendapatkan apa yang terbaik untuk dilakukan 10 menit yang lalu, ketika agen harus bertindak sekarang. Seringkali, sebaliknya, seorang *agent* harus menukar berapa lama waktu yang dibutuhkan untuk mendapatkan solusi dengan seberapa baik solusi tersebut; mungkin lebih baik menemukan solusi yang masuk

akal dengan cepat daripada menemukan solusi yang lebih baik nanti karena dunia akan berubah selama perhitungan.

Dimensi batas komputasi menentukan apakah suatu *agent* memiliki

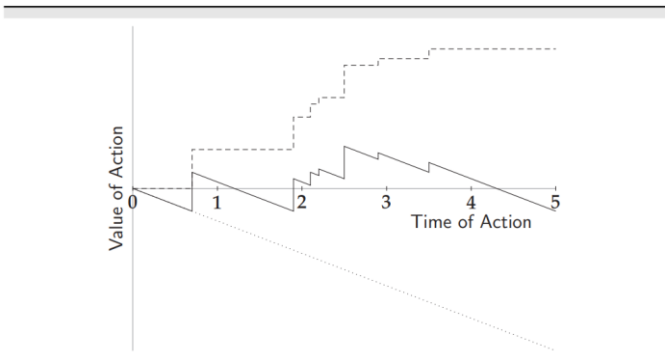
a. rasionalitas sempurna, di mana *agent* bernalar tentang tindakan terbaik tanpa memperhitungkan sumber daya komputasinya yang terbatas; atau
b. rasionalitas terbatas, di mana *agent* memutuskan tindakan terbaik yang dapat ditemukan mengingat keterbatasan komputasinya.

Batas sumber daya komputasi meliputi waktu komputasi, memori, dan akurasi numerik yang disebabkan oleh komputer yang tidak mewakili bilangan real dengan tepat.

Algoritma kapan saja adalah algoritme yang kualitas solusinya meningkat seiring waktu. Secara khusus, itu adalah salah satu yang dapat menghasilkan solusi terbaik saat ini kapan saja, tetapi dengan waktu yang lebih lama dapat menghasilkan solusi yang lebih baik lagi. Kami dapat memastikan bahwa kualitas tidak menurun dengan membiarkan agen menyimpan solusi terbaik yang ditemukan sejauh ini dan mengembalikannya saat dimintai solusi. Namun, menunggu untuk bertindak ada harganya; mungkin lebih baik *agent* bertindak sebelum menemukan solusi terbaik. Contoh : Gambar 2.3 menunjukkan bagaimana waktu komputasi suatu algoritma kapan saja dapat mempengaruhi kualitas solusi. *Agent* harus melakukan tindakan tetapi dapat melakukan beberapa perhitungan untuk memutuskan apa yang harus dilakukan. Kualitas solusi absolut, seandainya tindakan dilakukan pada waktu nol, ditunjukkan sebagai garis putus-putus di bagian atas, meningkat karena agen membutuhkan waktu untuk berpikir. Namun, ada hukuman yang terkait dengan

mengambil waktu untuk bertindak. Dalam gambar ini, hukumannya, ditampilkan sebagai garis putus-putus

di bagian bawah, sebanding dengan waktu yang dibutuhkan sebelum agen bertindak. Kedua nilai ini dapat ditambahkan untuk mendapatkan kualitas yang didiskon, nilai komputasi yang tergantung waktu; ini adalah garis padat di tengah grafik. Untuk contoh Gambar 1.5, seorang agen harus menghitung sekitar 2,5 satuan waktu, dan kemudian bertindak, pada titik mana kualitas yang didiskon mencapai nilai maksimumnya. Jika komputasi berlangsung lebih lama dari 4,3 unit waktu, kualitas solusi terdiskonto yang dihasilkan lebih buruk daripada jika algoritme hanya menampilkan tebakan awal yang dapat dihasilkannya tanpa komputasi. Biasanya kualitas solusi meningkat dalam lompatan; ketika solusi terbaik saat ini berubah, ada lompatan dalam kualitas. Namun, hukuman yang terkait dengan menunggu seringkali tidak sesederhana garis lurus.



Gambar 2.3 Waktu komputasi

Untuk mempertimbangkan rasionalitas terbatas, *agent* harus memutuskan apakah harus bertindak atau berpikir lebih banyak. Ini menantang karena seorang *agent* biasanya tidak tahu seberapa jauh lebih baik jika hanya menghabiskan sedikit lebih banyak waktu untuk berpikir. Selain itu, waktu yang dihabiskan untuk

memikirkan apakah itu harus bernalar dapat mengurangi penalaran yang sebenarnya tentang domain tersebut. Namun, rasionalitas terbatas dapat menjadi dasar penalaran perkiraan.

2.5 Aplikasi Prototipikal dari *Agent*

Aplikasi AI tersebar luas dan beragam dan mencakup diagnosis medis, penjadwalan proses pabrik, robot untuk lingkungan berbahaya, permainan game, kendaraan otonom di luar angkasa, sistem terjemahan bahasa alami, dan sistem bimbingan belajar. Daripada memperlakukan setiap aplikasi secara terpisah, kami mengabstraksi fitur-fitur penting dari aplikasi semacam itu untuk memungkinkan kami mempelajari prinsip-prinsip di balik penalaran dan tindakan yang cerdas.

Keempat domain aplikasi tersebut adalah sebagai berikut:

a. Robot pengiriman otonom berkeliaran di sekitar gedung mengantarkan paket dan kopi ke orang-orang di dalam gedung. *Agent* pengiriman ini harus dapat menemukan jalur, mengalokasikan sumber daya, menerima permintaan dari orang, membuat keputusan tentang prioritas, dan mengirimkan paket tanpa melukai orang atau dirinya sendiri.
b. Asisten diagnostik membantu manusia memecahkan masalah dan menyarankan perbaikan atau perawatan untuk memperbaiki masalah. Salah satu contohnya adalah asisten tukang listrik yang menyarankan apa yang mungkin salah di sebuah rumah, seperti saklar lampu putus, atau lampu padam, mengingat beberapa gejala masalah listrik. Contoh lain adalah seorang ahli diagnosa medis yang menemukan potensi penyakit, tes yang

berguna, dan perawatan yang tepat berdasarkan pengetahuan tentang domain medis tertentu dan gejala serta riwayat pasien. Asisten ini harus dapat menjelaskan alasannya kepada orang yang melakukan pengujian dan perbaikan dan yang pada akhirnya bertanggung jawab atas tindakannya.

c. Sebuah sistem bimbingan berinteraksi dengan siswa, menyajikan informasi tentang beberapa domain dan memberikan tes pengetahuan atau kinerja siswa. Ini memerlukan lebih dari menyajikan informasi kepada siswa. Melakukan apa yang dilakukan guru yang baik, yaitu menyesuaikan informasi yang disajikan kepada setiap siswa berdasarkan pengetahuannya, preferensi belajar. Sistem harus memahami materi pelajaran dan bagaimana siswa belajar.

d. Seorang *agent* perdagangan mengetahui apa yang diinginkan seseorang dan dapat membeli barang dan jasa atas namanya. Itu harus mengetahui persyaratan dan preferensinya dan bagaimana menukar tujuan yang bersaing. Misalnya, untuk liburan keluarga, agen perjalanan harus memesan hotel, penerbangan, persewaan mobil, dan hiburan, yang semuanya harus cocok satu sama lain. Ini harus menentukan trade-off pelanggan. Jika hotel yang paling cocok tidak dapat menampung keluarga sepanjang hari, harus ditentukan apakah mereka akan memilih untuk tinggal di hotel yang lebih baik untuk sebagian masa tinggal atau jika mereka memilih untuk tidak pindah hotel. Bahkan mungkin bisa berbelanja untuk mendapatkan barang spesial atau menunggu sampai penawaran bagus muncul.

2.6 *Monte Carlo Method*

Dikutip dari buku yang berjudul " *monte carlo methods* " oleh Barbu & Chun zu yang menjelaskan bahwa *monte carlo methods* merupakan sebuah pendekatan komputational yang menggunakan teknik pengambilan sampel acak untuk memecahkan masalah numerik yang sulit.

Monte carlo methods adalah sebuah pendekatan dalam *reinforcement learning* yang mengandalkan pengumpulan pengalaman dari interaksi langsung untuk mengestimasi nilai-nilai dan kebijakan yang optimal. Namun untuk menemukan nilai yang optimal setiap tindakan dari lingkungan harus dikunjungi. Pada metode *monte carlo* sebuah dinamika dari lingkungan akan diabaikan serta kebijakan optimal harus dipelajari oleh agen melalui interaksi *trial-error* dengan lingkungan.

Berikut merupakan persamaan *monte carlo method* untuk memperbarui *value function*.

$$V(S) \leftarrow V(S) + \alpha[G - V(S)]$$

Dimana V adalah value function, S adalah state pada waktu tertentu, alpha adalah step size, dan G adalah reward di akhir episode. Value function optimal V(S) akan cocok dengan real reward di setiap askhir episode untuk semua state. Untuk menentukan value function optimal, fungsi pembaruan diatas ditampilan.

monte carlo method memiliki 2 cara yang digunakan untuk menemukan nilai optimal sehingga agen harus memberikan tindakan pada lingkungan yang dikunjungi, yaitu *on policy* dan *off policy*. *On policy* kebijakan yang digunakan untuk menghasilkan episode yang akan dievaluasi dan diperbaiki. *Off policy*

memiliki 2 kebijakan satu untuk memilih tindakan dan satu lagi kebijakan target yang akan diperbaiki.

2.7 *Linear Function Approximation*

Dalam *reinforcement learning* pendekatan fungsi didasarkan pada data sampel yang dikumpulkan selama interaksi dengan lingkungan. Dalam ruang lingkungan yang besar, seringkali tidak praktis untuk mengunjungi atau mempresentasikan semua *state* dan tindakan, sehingga *function approximation* dapat digunakan untuk mempresentasikan kebijakan, fungsi nilai (Kober & Peter , 2012).

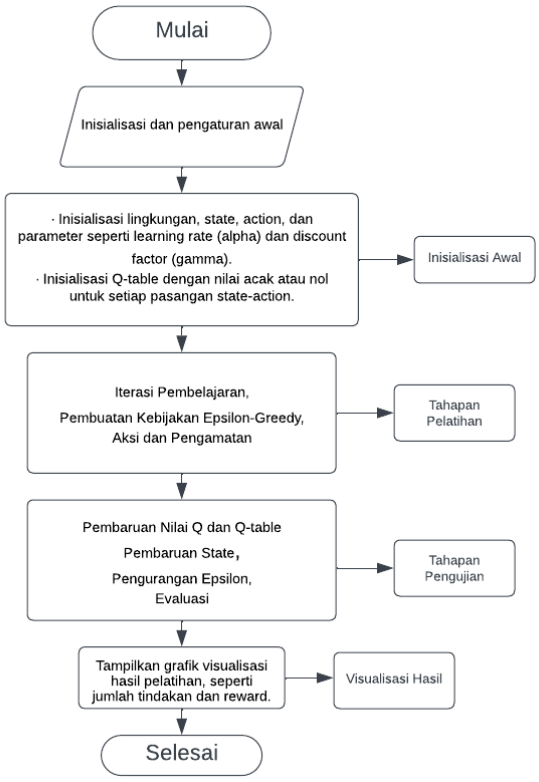
Algoritma *linear function approximation* adalah pendekatan dalam *deep reinforcement learning* yang memungkinkan agen untuk mengestimasi nilai-nilai state menggunakan fungsi linear dari fitur-fitur state. Ini berguna ketika ruang state sangat besar atau kontinu sehingga sulit atau tidak efisien untuk menyimpan nilai-nilai state dalam tabel.

Pada penelitian yang dilakukan oleh (huang et al,. 2014) yang menyatakan bahwa *linear function approximation* mempunyari 4 fitur ($\varphi 1$, $\varphi 2$, $\varphi 3$, $\varphi 4$) untuk lingkungan yang ditentukan sebagai contoh jarak horizontal dan vertikal dari agen ke tujuan untuk melewati rintangan.

BAB III
METODE PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian ini meliputi inisialisasi dan pengaturan awal, pelatihan *monte carlo* dan *linear function approximation,* pengujian *monte carlo* dan *linear function approximation,* dan visualisasi hasil. Diagram alir penelitian ditunjukkan pada Gambar 3.1 berikut ini.



Gambar 3.1 Diagram Alur Penelitian

Dalam konteks algoritma *monte carlo* dan *linear function approximation,* istilah-istilah tersebut memiliki arti berikut:

1. Agent: Agen atau *agent* komputer adalah entitas yang belajar dan mengambil tindakan dalam lingkungan. Dalam konteks ini, *agent* adalah entitas yang belajar melalui interaksi dengan lingkungan dan mengambil tindakan untuk mencapai tujuan yang telah ditetapkan.
2. Obstacle: Rintangan atau obstacle adalah elemen-elemen dalam lingkungan yang menghalangi pergerakan *agent*. Dalam implementasi ini, *agent* diharapkan dapat menghindari rintangan atau obstacle dalam perjalanan menuju tujuan.
3. Target: Target adalah tujuan yang ingin dicapai oleh *agent* dalam lingkungan. Agen berupaya mencapai target ini dengan mengambil tindakan yang tepat dan menghindari rintangan.
4. Environment: Environment adalah dunia di mana agen beroperasi. Ini adalah sumber interaksi utama bagi agen, dan environment dapat memiliki berbagai tingkat kompleksitas.
5. Actions: Actions (tindakan) adalah tindakan yang dapat diambil oleh agen dalam lingkungannya. Agen memilih tindakan berdasarkan strategi atau kebijakan yang telah dipelajari atau dikembangkan. Setiap tindakan yang diambil oleh agen dapat memengaruhi keadaan lingkungan.
6. Reward: Reward (imbalan) adalah umpan balik yang diberikan oleh lingkungan sebagai respons terhadap tindakan yang diambil oleh agen. Imbalan ini mengukur sejauh mana tindakan yang diambil oleh agen

mendukung atau menghambat mencapai tujuannya. Tujuannya adalah untuk memaksimalkan total imbalan yang diterima oleh agen seiring waktu.

7. State: State (keadaan) adalah representasi dari situasi atau kondisi saat ini dalam lingkungan. Keadaan menyediakan informasi tentang kondisi lingkungan saat ini yang diperlukan untuk membuat keputusan. Agen menggunakan informasi ini bersama dengan strateginya untuk memilih tindakan yang optimal.

Dalam *monte carlo* dan *linear function approximation*, agent berupaya bergerak menuju target sambil menghindari rintangan yang mungkin ada di sekitarnya. Tujuan utama agent adalah mencapai target dengan cara memaksimalkan total reward yang diperoleh selama proses perjalanan.

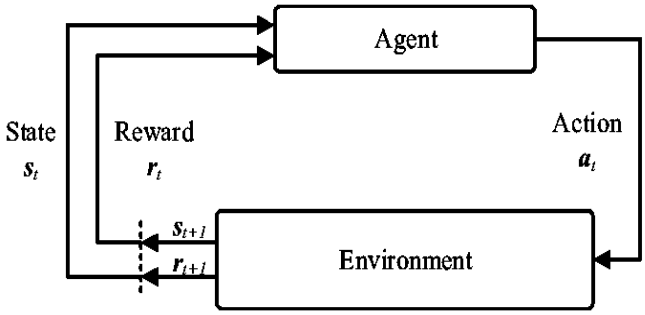
Pada penelitian ini, dilakukan variasi lingkungan dan variasi jumlah obstacle. Variasi lingkungan antara lain lingkungan berbentuk persegi, lingkungan berbentuk lingkaran, dan lingkungan berbentuk huruf. Sedangkan pada variasi jumlah obstacle terdapat tiga jumlah obstacle yang berbeda yaitu 1 obstcale, 2 obstacle, dan 3 obstacle.

3.2 Siklus Single Agent

Siklus *Single Agent monte carlo* dan *linear function approximation* adalah kerangka kerja yang digunakan dalam pembelajaran penguatan (reinforcement learning) ketika hanya ada satu agen (agent) yang berinteraksi dengan lingkungannya (environment). Siklus *Single Agent Reinforcement Learning* bekerja seperti berikut:

a. Agent mulai dalam suatu keadaan awal (initial state) dalam lingkungan.
b. Agent memutuskan tindakan yang akan diambil berdasarkan kebijakan yang dimilikinya.
c. Tindakan yang diambil oleh agen memengaruhi lingkungan, dan lingkungan merespons dengan memberikan imbalan kepada agen.
d. Agent mengamati keadaan baru yang dihasilkan oleh tindakannya dan memutuskan tindakan berikutnya.
e. Proses ini berlanjut selama agen berinteraksi dengan lingkungannya dalam serangkaian tindakan dan keadaan.
f. Tujuan agen adalah mempelajari kebijakan yang optimal, yang akan memungkinkannya untuk memaksimalkan akumulasi imbalan seiring waktu.

Selama siklus ini, agen terus belajar dari pengalaman dan mencoba meningkatkan kinerjanya dengan mengoptimalkan kebijakannya. Siklus *Single Agent monte carlo* dan *linear function approximation* ditunjukkan pada Gambar 3.2.



Gambar 3.2 Siklus Single Agent

3.3 Algoritma Pelatihan

Pada penelitian ini terdapat enam algoritma yang dapat diterapkan untuk melatih *agent* dalam pengambilan keputusan agar dapat berinteraksi dengan lingkungan *(environtment).* Algoritma yang dapat diterapkan adalah *monte carlo* dan *linear function approximation.*

3.3.1. *Monte Carlo Methods*

Monte Carlo Methods melakukan pendekatan dalam *reinforcement learning* dengan mengandalkan pengumpulan pengalaman dari interaksi langsung agent dengan lingkungan untuk mengestimasi nilai-nilai optimal dan kebijakan yang optimal. Berikut adalah langkah-langkah umum dari algoritma *Monte Carlo Methods* dalam *reinforcement learning single agent* (Mulvey, D. dkk, 2019):

1. Definisi Masalah
    a. Tetapkan lingkungan atau tugas yang ingin diselesaikan oleh agent.
    b. Tentukan state (keadaan) yang mungkin ada di lingkungan.
    c. Tetapkan action (tindakan) yang dapat diambil oleh agent di setiap state.
    d. Definisikan fungsi reward yang memberikan umpan balik positif atau negatif terkait tindakan yang diambil oleh agent.
2. Inisialisasi Tabel Nilai (Value Table) atau Kebijakan

    Buat tabel nilai (value table) untuk memetakan setiap state ke nilai-nilai awal atau kebijakan awal yang diambil oleh agent.

3. Pengumpulan Pengalaman (Episodes)

a. Mulai pengumpulan pengalaman dengan mengizinkan agent berinteraksi dengan lingkungan.
b. Agent melakukan tindakan berdasarkan kebijakan saat ini dan mengamati reward serta transisi ke state berikutnya.
c. Simulasikan episode (runtunan tindakan) hingga agent mencapai state terminal.

4. Perhitungan Return (Reward Kembali)
   a. Setelah agent mencapai state terminal, hitung nilai return (total reward yang diperoleh dari state tersebut hingga akhir episode).
   b. Return dihitung dengan menjumlahkan reward di setiap langkah episode berdasarkan faktor diskon (gamma), yaitu return = Σ(gamma^t * reward_t) untuk t=0 hingga T, dengan T adalah langkah terakhir dalam episode.
5. Update Nilai atau Kebijakan
   a. Setelah setiap episode selesai, update tabel nilai atau kebijakan berdasarkan return yang diperoleh dalam episode tersebut.
   b. Pada metode Monte Carlo FirstVisit, nilai atau kebijakan hanya diperbarui setelah agent mengunjungi suatu state untuk pertama kalinya dalam satu episode.
   c. Pada metode Monte Carlo EveryVisit, nilai atau kebijakan diperbarui setelah setiap kunjungan ke state dalam satu episode.
6. Konvergensi dan Evaluasi
   a. Ulangi langkah 3 hingga 5 sejumlah episode tertentu atau hingga nilai-nilai konvergen.

b. Evaluasi performa agent dengan mengukur rata-rata return atau reward yang diperoleh dari kebijakan yang dihasilkan.

Berikut adalah rumus-rumus yang digunakan dalam metode Monte Carlo:

1. Return (G)

Return adalah jumlah hadiah yang diterima oleh agen selama suatu episode. Dalam notasi matematisnya, return (G) dari waktu t dalam suatu episode adalah sebagai berikut:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T$$

Di sini, $R_{t+1} + R_{t+2} + \cdots + R_T$ adalah reward yang diterima oleh agen dari waktu $t + 1$ hingga waktu T, di mana T adalah waktu akhir episode.

2. State-Value (V(s))

State-value (nilai keadaan) adalah estimasi nilai dari keadaan s yang mengukur harapan return yang diterima agen ketika memulai dari keadaan s dan mengikuti kebijakan tertentu. State-value dalam MC diperkirakan sebagai rata-rata dari semua pengembalian yang diamati ketika agen berada dalam keadaan s:

$$V(s) = \mathbb{E}[G_t|S_t = s]$$

Di sini, $\mathbb{E}$ adalah operator ekspektasi.

3. Action-Value $(Q(s, a))$

Action-value (nilai aksi) adalah estimasi nilai dari tindakan a yang dilakukan dalam keadaan s. Action-value dalam MC diperkirakan sebagai rata-rata dari semua pengembalian yang diamati ketika agen berada dalam keadaan s, mengambil tindakan a, dan kemudian mengikuti kebijakan tertentu:

$$(Q(s, a)) = \mathbb{E}[G_t|S_t = s, A_t = a]$$

Di sini, $S_t = s$ dan $A_t = a$ menunjukkan bahwa agen berada dalam keadaan s dan mengambil tindakan a pada waktu t.

Dalam Monte Carlo, estimasi nilai-nilai ($V(s)$ atau $Q(s, a)$) diperoleh dengan mengambil rata-rata pengembalian yang diterima selama banyak episode yang telah diamati sebelumnya. Semakin banyak episode yang diamati, semakin baik estimasi nilai-nilai tersebut. MC memerlukan pengumpulan pengembalian dari seluruh episode untuk melakukan estimasi yang akurat. Setelah estimasi nilai-nilai diperoleh, agen dapat mengoptimalkan kebijakan berdasarkan nilai-nilai ini.

3.3.2. *Linear Function Approximation (LFA)*

Algoritma *Linear Function Approximation (LFA)* melakukan pendekatan dalam *reinforcement learning* dengan memungkinkan agent untuk mengestimasi nilai-nilai state menggunakan fungsi linear dari fitur-fitur state. Ini berguna ketika ruang state sangat besar atau kontinu sehingga sulit atau tidak efisien untuk menyimpan nilai-nilai state dalam tabel. Berikut adalah langkah-langkah umum dari algoritma *Linear Function Approximation* dalam *reinforcement learning single agent* (Mulvey, D. dkk, 2019)*:*

1. Definisi Masalah
    a. Tetapkan lingkungan atau tugas yang ingin diselesaikan oleh agent.
    b. Tentukan fitur-fitur yang relevan dari state yang akan digunakan dalam aproksimasi fungsi nilai.
2. Inisialisasi Bobot (Weights)

a. Inisialisasi bobot (weights) dari fungsi aproksimasi nilai secara acak atau dengan nilai-nilai awal tertentu.

3. Pengumpulan Pengalaman (Episodes)

   a. Mulai pengumpulan pengalaman dengan mengizinkan agent berinteraksi dengan lingkungan.
   b. Agent melakukan tindakan berdasarkan kebijakan saat ini dan mengamati reward serta transisi ke state berikutnya.
   c. Simulasikan episode (runtunan tindakan) hingga agent mencapai state terminal.

4. Perhitungan Fungsi Aproksimasi

   a. Dalam setiap state, hitung nilai aproksimasi menggunakan fungsi linear dan bobot yang diinisialisasi
   b. nilai aproksimasi = bobot^T * fitur-fitur state
   c. ^T menunjukkan operasi transposisi pada vektor bobot.

5. Perhitungan Error dan Update Bobot

   a. Setelah perhitungan nilai aproksimasi, hitung error antara nilai aproksimasi dan nilai aktual (TD error).
   b. TD error = reward + gamma * nilai aproksimasi state berikutnya nilai aproksimasi state saat ini
   c. Gunakan TD error untuk mengupdate bobot (weights) dengan metode gradient descent atau metode lainnya
   d. bobot baru = bobot lama + alpha * TD error * fitur-fitur state

   Alpha adalah learning rate yang mengontrol sejauh mana perubahan dalam bobot yang diperbolehkan setiap kali update.

6. Konvergensi dan Evaluasi

a. Ulangi langkah 3 hingga 5 sejumlah episode tertentu atau hingga nilai-nilai konvergen.
b. Evaluasi performa agent dengan mengukur rata-rata reward yang diperoleh dari kebijakan yang dihasilkan.

Rumus-rumus yang digunakan dalam Linear Function Approximation adalah sebagai berikut:

1. Value Function Approximation ($V(s)$)

Dalam kasus ini, agen memperkirakan nilai dari suatu keadaan $V(s)$ sebagai kombinasi linier dari fitur-fitur keadaan $x(s)$ dengan bobot w yang harus dipelajari:

$$V(s) \approx w^T x(s)$$

Di sini, $x(s)$ adalah vektor fitur keadaan yang mewakili keadaan s, dan w adalah vektor bobot yang harus disesuaikan agar perkiraan sesuai dengan nilai sebenarnya.

2. Action-Value Function Approximation ($Q(s,a)$)

Dalam kasus ini, agen memperkirakan nilai dari suatu tindakan $Q(s,a)$ sebagai kombinasi linier dari fitur-fitur tindakan $x(s,a)$ dengan bobot w:

$$Q(s,a) \approx w^T x(s,a)$$

Di sini, $x(s,a)$ adalah vektor fitur tindakan yang mewakili tindakan a dalam keadaan s.

BAB IV
HASIL DAN PEMBAHASAN

Pada penelitian ini, *single agent monte carlo* dan *linear function approximation* diimplementasikan untuk mengajari *agent* menghindari *obstacle* dan mencapai target. Beberapa algoritma diterapkan untuk melatih *agent* dalam pengambilan keputusan agar dapat berinteraksi dengan lingkungan *(environtment).* Algoritma yang diterapkan antara lain: *monte carlo* dan *linear function approximation*

4.1. Penentuan koordinat awal

Untuk melihat pengaruh letak *obstacle* terhadap perjalanan *agent* menuju *target,* koordinat awal *obstacle* divariasikan sebanyak tiga kali. Koordinat awal *agent, obstacle, dan target* ditunjukkan pada Tabel 4.1 berikut ini.

Tabel 4.1 Koordinat *awal* agent, obstacle, dan target

Variasi	Koordinat Awal		
	Agent	*Obstacle*	*Target*
1	(5,5)	(10,10)	(25,25)
2		(15,15)	
3		(20,20)	

4.2 Implementasi Algoritma *Monte Carlo Method*

Metode *Monte Carlo* digunakan untuk memperkirakan nilai-nilai optimal dan kebijakan optimal dalam *reinforcement learning* tanpa mengharuskan adanya informasi penuh tentang lingkungan dan tanpa menggunakan model lingkungan. Metode ini memperkirakan nilai-nilai optimal dengan menghitung rerata imbalan

yang diperoleh dari jalur (trajectory) yang diambil agen dalam lingkungan. Metode *Monte Carlo* dibagi menjadi dua jenis yaitu *On-Policy Monte Carlo* dan *Off-Policy Monte Carlo.*

4.2.1 **On-Policy Monte Carlo**

On-policy monte carlo bertujuan untuk memperbarui dan meningkatkan kebijakan agen berdasarkan pengalaman nyata yang diperoleh dari interaksi dengan lingkungan menggunakan kebijakan yang sama. *On-Policy Monte Carlo* bekerja dengan menghasilkan episode-episode berdasarkan kebijakan yang digunakan oleh agen. Dalam setiap episode, agen berinteraksi dengan lingkungan dan mencatat tindakan yang diambil dan imbalan yang diterima. Episode-episode ini digunakan untuk memperkirakan nilai-nilai state dan state-action pair serta memperbarui kebijakan.

Langkah-langkah algoritma *On-Policy Monte Carlo*

1. Inisialisasi:
    a. Menginisialisasi lingkungan (`env`) menggunakan kelas `SAEnvironment`.
    b. Variabel `isTraining` digunakan untuk menentukan apakah kode berada dalam mode pelatihan atau hanya menampilkan hasil yang sudah ada.
2. Inisialisasi Parameter:
    a. Jika dalam mode pelatihan (`isTraining = true`), beberapa parameter penting diatur:
    b. `NUM_ITERATIONS`: Jumlah iterasi (episode) yang akan dilakukan.
    c. `gamma`: Faktor diskon untuk menghitung return.

d. `epsilon`: Probabilitas eksplorasi, yang mengontrol seberapa sering agen mengambil tindakan acak.
e. `min_epsilon`: Batas bawah untuk probabilitas eksplorasi.
f. `Q`: Matriks nilai tindakan (Q-values).
g. `Returns`: Matriks yang menyimpan jumlah return dan hitungan untuk setiap state-action pair.
h. `Policy`: Matriks kebijakan awal dengan probabilitas yang merata.

3. Loop Iterasi (Episode):
   a. Loop utama yang akan berjalan sebanyak `NUM_ITERATIONS` kali.
   b. Pada setiap iterasi (episode), lingkungan di-reset dan agen mulai berinteraksi dengan lingkungan.

4. Iterasi dalam Episode:
   a. Agen melakukan interaksi dengan lingkungan selama episode, memilih tindakan berdasarkan kebijakan yang digunakan.
   c. Imbalan yang diterima dan tindakan yang diambil dicatat dalam matriks `matrix_episode`.

5. Pembaruan dan Perhitungan Return:

Setelah episode selesai, agen memeriksa setiap langkah yang diambil dalam episode dan menghitung return untuk setiap state-action pair yang belum pernah dikunjungi sebelumnya.

6. Perbaruan Nilai dan Kebijakan:

Untuk setiap state-action pair yang belum pernah dikunjungi dalam episode, agen menghitung return dan memperbarui nilai tindakan (`Q`) serta kebijakan (`Policy`) berdasarkan return yang dihitung.

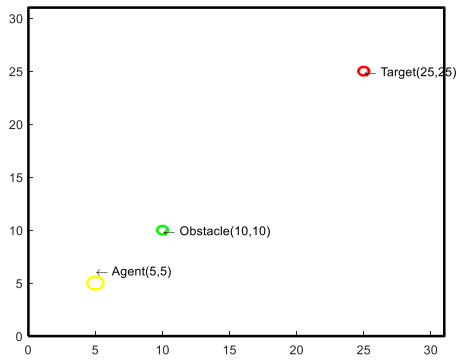
7. Penyimpanan dan Visualisasi:

   a. Jika dalam mode pelatihan, hasil dari iterasi disimpan dalam berbagai file (seperti `onpmc_q.mat`, `onpmc_returns.mat`, dll.).

   b. Visualisasi yang menunjukkan jumlah tindakan dan imbalan yang diperoleh dari setiap iterasi.

Skenario 1

Pada skenario 1, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (10,10), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.1.



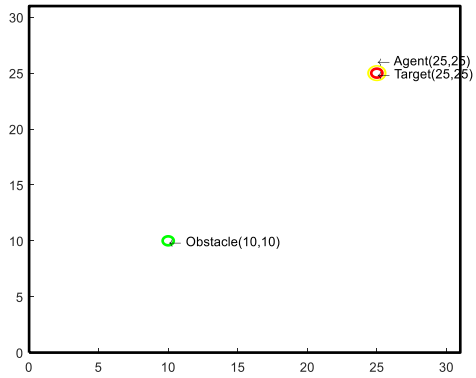
Gambar 4.1 koordinat awal masing-masing objek pada skenario 1

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (10,10). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.2 berikut ini.

Tabel 4.2 Jumlah action dan reward pada skenario 1

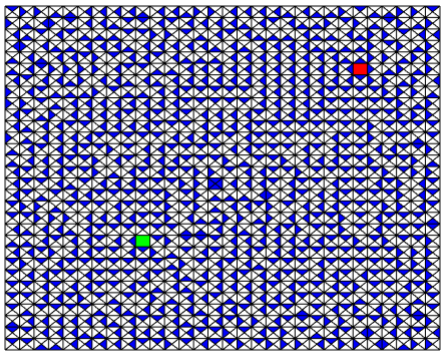
Iteration	Actions Taken	Final Reward
1	14	-1
2	60	1
3	16	-1
4	46	1
5	62	1

Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.1.



Gambar 4.2 koordinat akhir masing-masing objek pada skenario 1

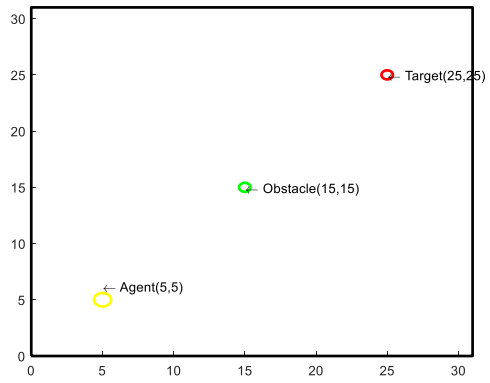
Tabel 4.2 dan Gambar 4.1 menunjukkan bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi, namun mengalami 2 kali tabrakan terhadap obstacle. Perjalanan tercepat terjadi pada iterasi pertama dengan jumlah *action* sebanyak 14. Kebijakan dari *On-Policy Monte Carlo* untuk skenario 1 ditunjukkan pada Gambar 4.3.



Gambar 4.3 Kebijakan dari On-Policy Monte Carlo untuk skenario 1

Skenario 2

Pada Skenario 2, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (15,15), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.4.



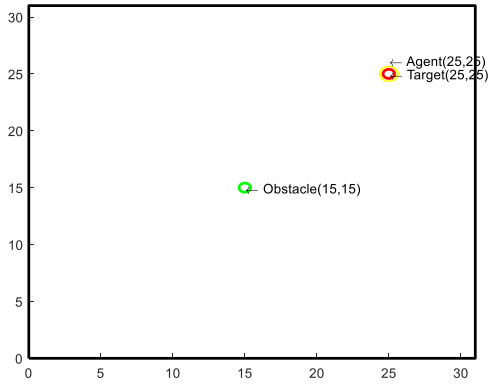
Gambar 4.4 koordinat awal masing-masing objek pada Skenario 2

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (15,15). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.3 berikut ini.

Tabel 4.3 Jumlah action dan reward pada Skenario 2

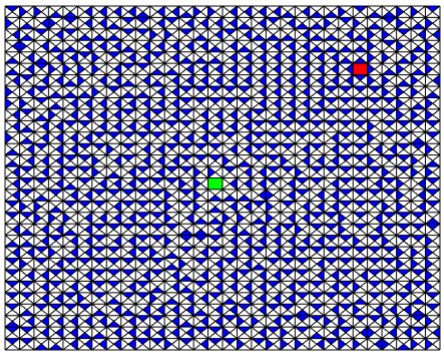
Iteration	Actions Taken	Final Reward
1	52	1
2	48	1
3	56	1
4	58	1
5	52	1

Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.5.



Gambar 4.5 koordinat akhir masing-masing objek pada skenario 2

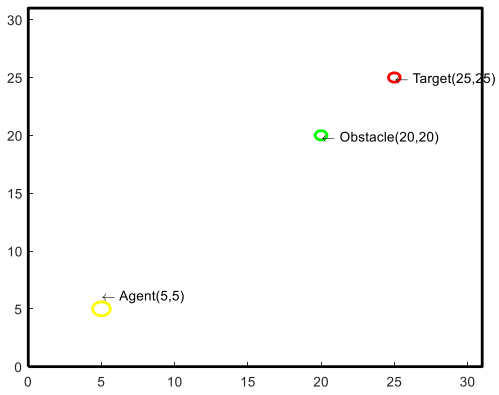
Tabel 4.3 dan Gambar 4.5 menunjukkan bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi. Perjalanan tercepat terjadi pada iterasi ke-dua dengan jumlah *action* sebanyak 48. Kebijakan dari *On-Policy Monte Carlo* untuk skenario 2 ditunjukkan pada Gambar 4.6.



Gambar 4.6 Kebijakan dari On-Policy Monte Carlo untuk skenario 2

Skenario 3

Pada skenario 3, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (20,20), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.7.



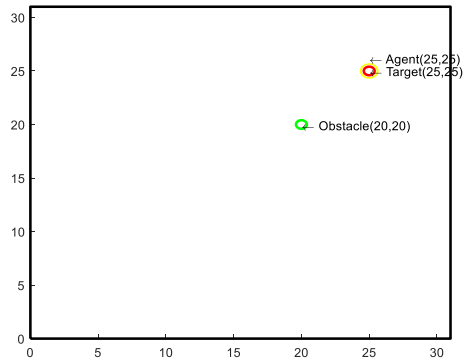
Gambar 4.7 koordinat awal masing-masing objek pada skenario 3

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (20,20). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.4 berikut ini.

Tabel 4.4 Jumlah action dan reward pada skenario 3

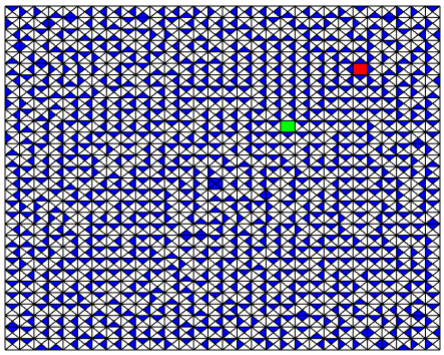
Iteration	Actions Taken	Final Reward
1	52	1
2	48	1
3	56	1
4	58	1
5	52	1

Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.8.



Gambar 4.8 koordinat akhir masing-masing objek pada skenario 3

Tabel 4.4 dan Gambar 4.8 menunjukkan bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi. Perjalanan tercepat terjadi pada iterasi ke-dua dengan jumlah *action* sebanyak 48. Kebijakan dari *On-Policy Monte Carlo* untuk skenario 3 ditunjukkan pada Gambar 4.9.



Gambar 4.9 Kebijakan dari On-Policy Monte Carlo untuk skenario 3

Berdasarkan percobaan yang telah dilakukan, algoritma *On-Policy Monte Carlo* berhasil membantu agen memperbarui kebijakan dan memperkirakan nilai-nilai optimal berdasarkan pengalaman nyata yang diperoleh dari interaksi dengan lingkungan. Algoritma ini berfokus pada pembelajaran *on-policy*, di mana agen menggunakan kebijakan yang sedang diperbarui untuk mengambil tindakan selama interaksi.

4.2.2 Off-Policy Monte Carlo

Tujuan utama dari *Off-Policy Monte Carlo* adalah memperbarui kebijakan dan memperkirakan nilai-nilai optimal berdasarkan pengalaman yang diperoleh dari kebijakan eksplorasi yang berbeda dengan kebijakan target yang ingin dipelajari. *Off-Policy Monte Carlo* memisahkan antara kebijakan eksplorasi *(epsilon-greedy)* dan kebijakan target yang ingin dipelajari. Pengalaman dikumpulkan dengan menjalankan kebijakan eksplorasi untuk mengumpulkan data dan kemudian menggunakannya untuk memperbarui kebijakan target.

Langkah-langkah algoritma *Off-Policy Monte Carlo*

1. Inisialisasi:
    a. Menginisialisasi lingkungan (`env`) menggunakan kelas `SAEnvironment`.
    b. Variabel `isTraining` digunakan untuk menentukan apakah kode berada dalam mode pelatihan atau hanya menampilkan hasil yang sudah ada.
2. Inisialisasi Parameter:
    a. Jika dalam mode pelatihan (`isTraining = true`), beberapa parameter penting diatur:
    b. `NUM_ITERATIONS`: Jumlah iterasi (episode) yang akan dilakukan.
    c. `gamma`: Faktor diskon untuk menghitung return.
    d. `epsilon`: Probabilitas eksplorasi untuk kebijakan eksplorasi.
    e. `min_epsilon`: Batas bawah untuk probabilitas eksplorasi.
    f. `Q`: Matriks nilai tindakan (Q-values).
    g. `C`: Matriks yang menyimpan kumulatif denominasi dari rumus importance sampling berbobot.
    h. `Policy`: Matriks kebijakan awal dengan probabilitas yang merata.
    i. Variabel lain untuk melacak iterasi dan imbalan.
3. Loop Iterasi (Episode):
    a. Loop utama yang akan berjalan sebanyak `NUM_ITERATIONS` kali.
    b. Pada setiap iterasi (episode), lingkungan di-reset dan agen mulai berinteraksi dengan lingkungan.
4. Iterasi dalam Episode:

Agen melakukan interaksi dengan lingkungan selama episode, memilih tindakan berdasarkan kebijakan eksplorasi dan kemudian kebijakan target.

5. Pembaruan dan Perhitungan Return:

Setelah episode selesai, agen memeriksa setiap langkah yang diambil dalam episode dan menghitung return untuk setiap langkah, menghitung G (return kumulatif) dan W (rasio importance sampling).

6. Perbaruan Nilai dan Kebijakan:

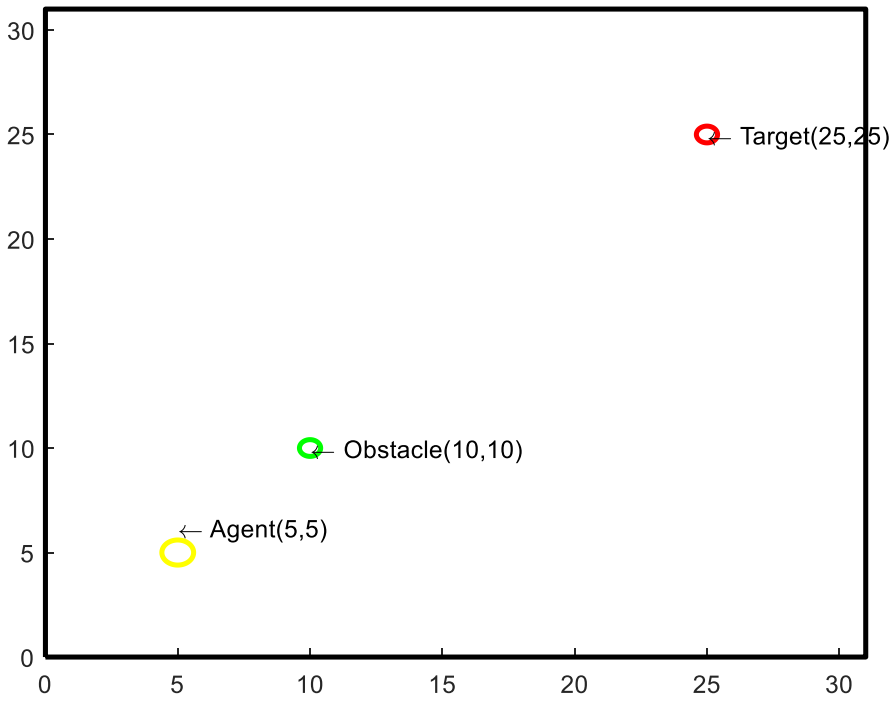
a. Untuk setiap langkah dalam episode, Q-value diperbarui berdasarkan rumus incremental update, di mana Q-value diperbarui dengan mempertimbangkan W dan return yang dihitung.
b. Kebijakan target diperbarui menjadi greedy berdasarkan Q-values yang diperbarui.

7. Penyimpanan dan Visualisasi:

a. Jika dalam mode pelatihan, hasil dari iterasi disimpan dalam berbagai file (seperti `offpmc_q.mat`, `offpmc_c.mat`, dll.).
b. Visualisasi yang menunjukkan jumlah tindakan dan imbalan yang diperoleh dari setiap iterasi.

Skenario 1

Pada skenario 1, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (10,10), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.10.



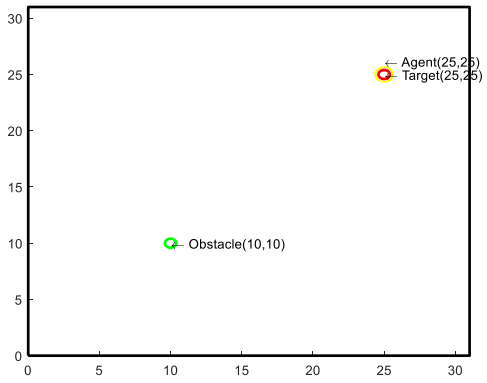
Gambar 4.10 koordinat awal masing-masing objek pada skenario 1

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (10,10). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.5 berikut ini.

Tabel 4.5 Jumlah action dan reward pada skenario 1

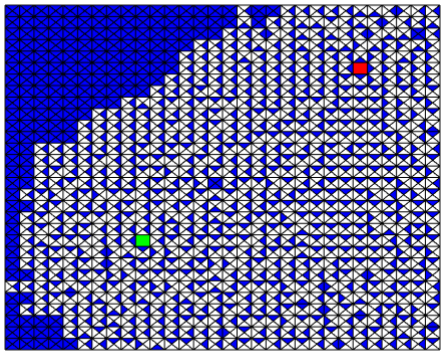
Iteration	Actions Taken	Final Reward
1	70	1
2	70	1
3	60	1
4	72	1
5	64	1

Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.11.



Gambar 4.11 koordinat akhir masing-masing objek pada skenario 1

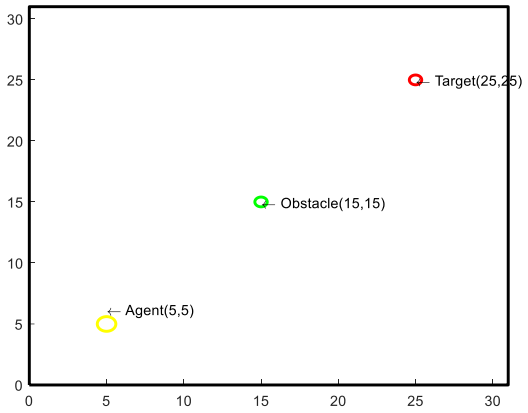
Tabel 4.5 dan Gambar 4.11 menunjukkan bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi. Perjalanan tercepat terjadi pada iterasi ke-tiga dengan jumlah *action* sebanyak 60. Kebijakan dari *Off-Policy Monte Carlo* untuk variasi 1 ditunjukkan pada Gambar 4.12.



Gambar 4.12 Kebijakan dari Off-Policy Monte Carlo untuk skenario 1

Skenario 2

Pada skenario 2, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (15,15), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.13.



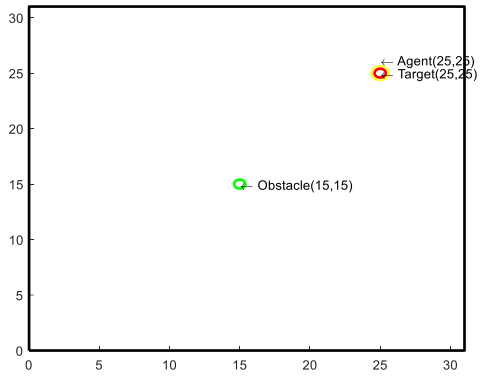
Gambar 4.13 koordinat awal masing-masing objek pada skenario 2

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (15,15). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.6 berikut ini.

Tabel 4.6 Jumlah action dan reward pada skenario 2

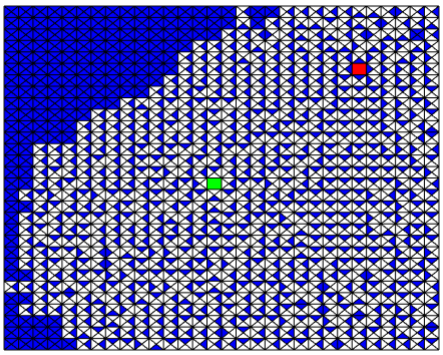
Iteration	Actions Taken	Final Reward
1	70	1
2	70	1
3	60	1
4	72	1
5	64	1

Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.14.



Gambar 4.14 koordinat akhir masing-masing objek pada skenario 2

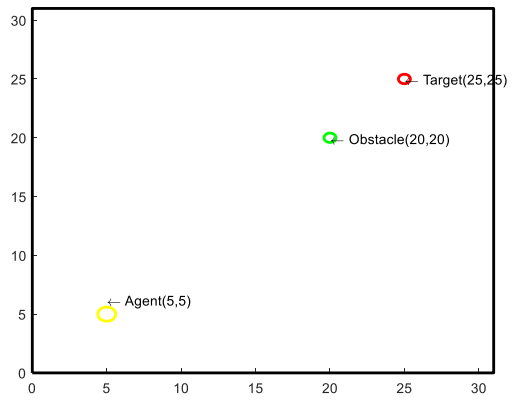
Tabel 4.6 dan Gambar 4.14 menunjukkan bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi. Perjalanan tercepat terjadi pada iterasi ke-tiga dengan jumlah *action* sebanyak 60. Kebijakan dari *Off-Policy Monte Carlo* untuk variasi 2 ditunjukkan pada Gambar 4.15.



Gambar 4.15 Kebijakan dari Off-Policy Monte Carlo untuk skenario 2

Skenario 3

Pada skenario 3, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (20,20), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.16.



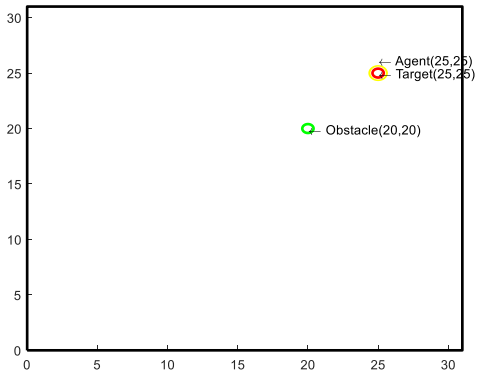
Gambar 4.16 koordinat awal masing-masing objek pada skenario 3

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (20,20). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.7 berikut ini.

Tabel 4.7 Jumlah action dan reward pada variasi 3

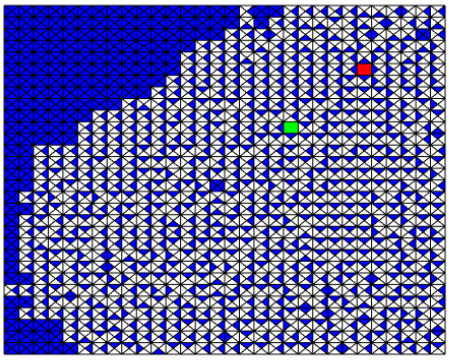
Iteration	Actions Taken	Final Reward
1	70	1
2	70	1
3	48	-1
4	64	1
5	50	1

Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.17.



Gambar 4.17 koordinat akhir masing-masing objek pada skenario 3

Tabel 4.7 dan Gambar 4.17 menunjukkan bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi, namun melakukan 1 kali tabrakan terhadap obstacle. Perjalanan tercepat terjadi pada iterasi ke-tiga dengan jumlah *action* sebanyak 48. Kebijakan dari *Off-Policy Monte Carlo* untuk variasi 3 ditunjukkan pada Gambar 4.18.



Gambar 4.18 Kebijakan dari Off-Policy Monte Carlo untuk skenario 3

Berdasarkan percobaan yang telah dilakukan, algoritma *Off-Policy Monte Carlo* berhasil membantu agen untuk memperbarui kebijakan dan memperkirakan

nilai-nilai optimal berdasarkan pengalaman yang diperoleh dari kebijakan eksplorasi dan kebijakan target yang berbeda. *Off-Policy Monte Carlo* memanfaatkan konsep importance sampling untuk menghubungkan dua kebijakan tersebut.

Pendekatan model-free Monte Carlo memungkinkan agen untuk belajar melalui pengalaman interaksi langsung dengan lingkungan, tanpa memerlukan pengetahuan tentang model lingkungan. Metode ini mengambil pendekatan berbasis eksperimen, di mana agen mengumpulkan sampel tindakan dan reward dari interaksi dengan lingkungan untuk memperkirakan nilai tindakan dan keadaan.

Metode Monte Carlo mengelompokkan hasil dari satu episode lengkap, mengumpulkan semua tindakan dan reward yang terkait dengan episode tersebut. Ini memungkinkan agen untuk belajar dari hasil yang lebih akurat dan terdistribusi dalam waktu. Metode ini cocok untuk menangani masalah *"reward delay,"* di mana reward mungkin terjadi setelah beberapa tindakan dilakukan. Hal ini karena metode Monte Carlo mencatat semua tindakan yang terjadi dalam satu episode sebelum memperbarui perkiraan nilai.

Metode Monte Carlo lebih cocok untuk lingkungan episodik, di mana interaksi agen dengan lingkungan terdiri dari episode yang terbatas. Dalam lingkungan kontinu atau semi-kontinu, metode ini mungkin memerlukan lebih banyak episode atau beberapa penyesuaian.

Metode Monte Carlo dapat memiliki konvergensi yang lambat terutama dalam lingkungan dengan episode yang panjang. Ini karena perkiraan nilai tindakan dan keadaan diperbarui hanya setelah akhir episode.

Seperti halnya metode reinforcement learning lainnya, dalam penerapan Monte Carlo, agen harus mengelola trade-off antara eksplorasi (mencoba tindakan baru) dan eksploitasi (memilih tindakan terbaik berdasarkan informasi saat ini). Metode Monte Carlo dapat diterapkan dalam berbagai jenis lingkungan dan tugas, termasuk masalah permainan atau situasi pengambilan keputusan yang kompleks.

Agar pembelajaran Monte Carlo efektif, agen mungkin perlu mengumpulkan banyak episode untuk mendapatkan perkiraan yang akurat dan stabil tentang nilai tindakan dan keadaan. Secara keseluruhan, metode Monte Carlo adalah pendekatan penting dalam reinforcement learning yang memungkinkan agen untuk belajar tanpa memerlukan pengetahuan tentang model lingkungan. Meskipun memiliki beberapa keterbatasan, metode ini dapat memberikan hasil yang baik dalam lingkungan yang sesuai dan dengan pengaturan yang tepat.

4.3 Implementasi Algoritma Linear Function Approximation

Linear Function Approximation (LFA) bertujuan untuk mengatasi masalah dimensi tinggi dari keadaan (state) dan tindakan (action) dalam lingkungan yang kompleks. Dalam konteks reinforcement learning untuk single-agent, tujuan utama LFA adalah untuk mengaproksimasi fungsi nilai (seperti Q-value atau nilai state) dengan menggunakan kombinasi linear dari fitur-fitur yang diambil dari keadaan dan tindakan.

Langkah-langkah Algoritma *Linear Function Approximation*

1. Inisialisasi Lingkungan:

`env = SAEnvironment;`: Objek lingkungan dibuat menggunakan kelas `SAEnvironment`.

2. Pengaturan Parameter:

a. `alpha = 0.1;`: Tingkat pembelajaran (learning rate) untuk algoritma LFA.

b. `gamma = 0.9;`: Faktor diskon untuk menilai pengaruh imbalan di masa depan.

c. `maxItr = 3000;`: Jumlah maksimum iterasi untuk mengakhiri satu episode.

d. `estimator = LFAEstimator(env,alpha);`: Membuat objek estimator LFA dengan parameter lingkungan dan tingkat pembelajaran.

3. Inisialisasi Mode Training:

Jika `isTraining` bernilai `true`, variabel berikut diinisialisasi:

1. `NUM_ITERATIONS`: Jumlah iterasi pelatihan.
2. `epsilon`: Nilai ε dalam kebijakan ε-greedy.
3. `min_epsilon`: Nilai minimum untuk ε.
4. `iterationCount(NUM_ITERATIONS)`: Array untuk menyimpan jumlah langkah dalam setiap iterasi.
5. `rwd(NUM_ITERATIONS)`: Array untuk menyimpan hadiah dalam setiap iterasi.

4. Inisialisasi Mode Pengujian:

Jika `isTraining` dan `isTesting` bernilai `false`, variabel berikut diinisialisasi:

1. `NUM_ITERATIONS`: Jumlah iterasi pengujian.

2. `iterationCount(NUM_ITERATIONS)`: Array untuk menyimpan jumlah langkah dalam setiap iterasi pengujian.
3. `rwd(NUM_ITERATIONS)`: Array untuk menyimpan hadiah dalam setiap iterasi pengujian.
4. Memuat bobot LFA yang telah dilatih sebelumnya dari file 'onp_lfa_weights.mat'.
5. Menetapkan bobot tersebut pada objek estimator dengan `estimator.set_weights(Weights)`.

5. Loop Utama:
   a. Loop ini akan dilakukan sebanyak `NUM_ITERATIONS` kali.
   b. Lingkungan direset ke posisi awal.
   c. Jika bukan mode pelatihan atau pengujian, lingkungan ditampilkan menggunakan `env.render()`.
6. Eksplorasi dan Eksplotasi:
   a. Mengambil tindakan awal menggunakan kebijakan ε-greedy.
   b. Memperkirakan nilai dari tindakan tersebut menggunakan estimator LFA.
7. Loop Episode:
   a. Loop ini berjalan selama episode belum selesai (`done = false`).
   b. Dalam loop episode, agen berinteraksi dengan lingkungan:
      1. Jika batas maksimum iterasi tercapai, episode dihentikan.
      2. Mengambil tindakan menggunakan kebijakan ε-greedy.
      3. Melakukan tindakan dan mendapatkan hasil berikutnya dari lingkungan.

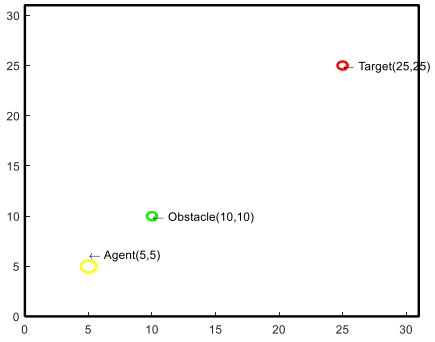
4. Jika bukan mode pelatihan, lingkungan ditampilkan.
5. Menghitung dan memperbarui nilai Q menggunakan algoritma On-Policy LFA.

8. Penyimpanan dan Visualisasi:

a. Jika dalam mode pelatihan, bobot estimator LFA disimpan dalam file 'onp_lfa_weights.mat'.
b. Data jumlah langkah dan hadiah disimpan dalam file terpisah.
c. Grafik batang jumlah langkah dan hadiah ditampilkan jika dalam mode pelatihan atau pengujian.

Skenario 1

Pada skenario 1, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (10,10), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.19.



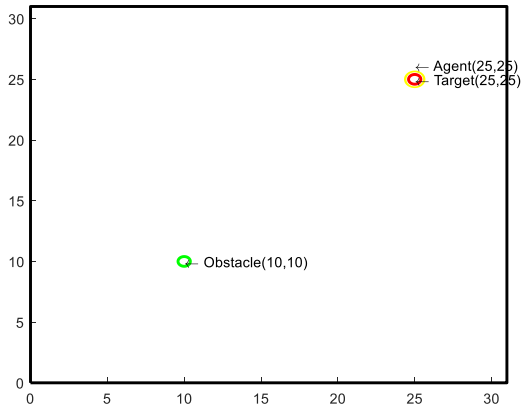
Gambar 4.19 koordinat awal masing-masing objek pada skenario 1

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (10,10). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.8 berikut ini.

Tabel 4.8 Jumlah action dan reward pada skenario 1

Iteration	Actions Taken	Final Reward
1	82	1
2	72	1
3	83	1
4	85	1
5	81	1

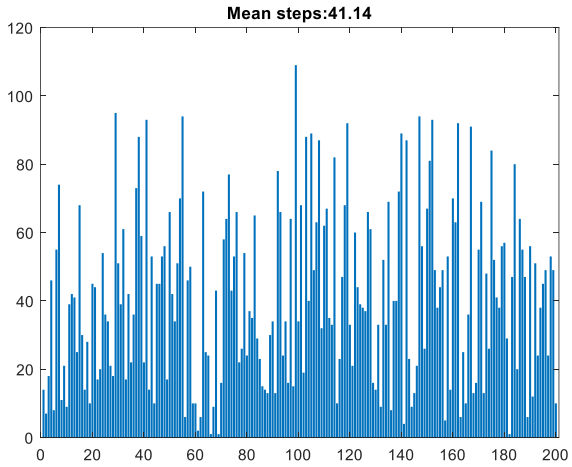
Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.20.



Gambar 4.20 koordinat akhir masing-masing objek pada skenario 1

Berdasarkan pada Tabel 4.8 dan Gambar 4.20, tampak bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi. Perjalanan tercepat terjadi pada iterasi ke-dua dengan jumlah *action* sebanyak 72.

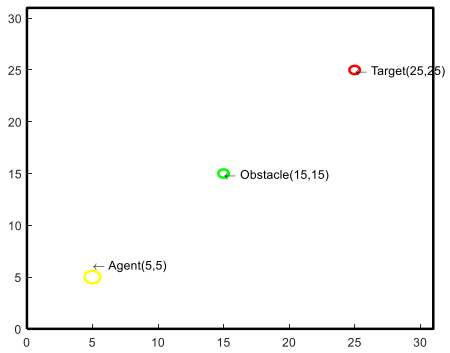
Distribusi jumlah langkah untuk mencapai tujuan dalam setiap episode ditunjukkan pada Gambar 4.21.



Gambar 4.21 Distribusi jumlah langkah untuk mencapai tujuan pada skenario 1

skenario 2

Pada variasi 2, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (15,15), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.22.



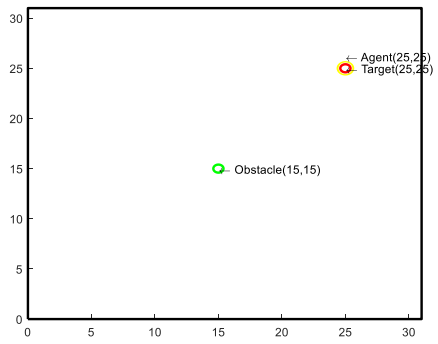
Gambar 4.22 koordinat awal masing-masing objek pada skenario 2

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (15,15). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.9 berikut ini.

Table 4.9 Jumlah action dan reward pada skenario 2

Iteration	Actions Taken	Final Reward
1	73	1
2	66	1
3	87	1
4	84	1
5	93	1

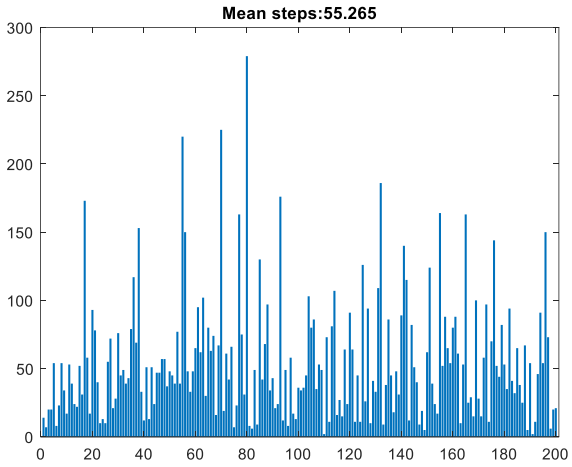
Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.23.



Gambar 4.23 koordinat akhir masing-masing objek pada skenario 2

Berdasarkan pada Tabel 4.9 dan Gambar 4.23, tampak bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi. Perjalanan tercepat terjadi pada iterasi ke-dua dengan jumlah *action* sebanyak 66.

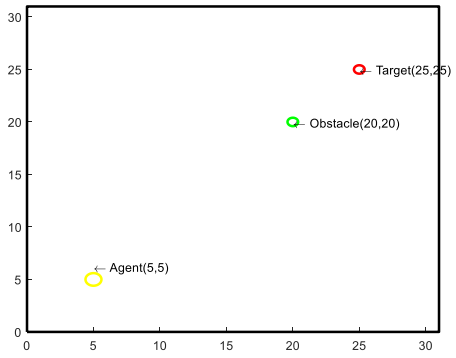
Distribusi jumlah langkah untuk mencapai tujuan dalam setiap episode ditunjukkan pada Gambar 4.24.



Gambar 4.24 Distribusi jumlah langkah untuk mencapai tujuan pada skenario 2

skenario 3

Pada variasi 3, koordinat awal untuk *Agent* adalah (5,5), koordinat awal untuk *Obstacle* adalah (20,20), dan koordinat awal untuk *Target* adalah (25,25). Gambaran koordinat awal masing-masing objek ditunjukkan pada Gambar 4.25.



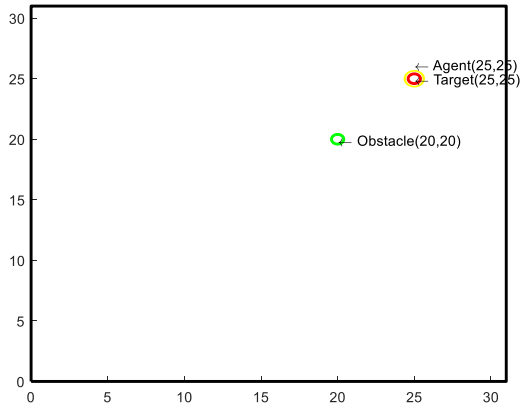
Gambar 4.25 koordinat awal masing-masing objek pada skenario 3

Agent (5,5) melakukan perjalanan menuju *target* (25,25) dengan menghindari *obstacle* (20,20). Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.10 berikut ini.

Tabel 4.10 Jumlah action dan reward pada skenario 3

Iteration	Actions Taken	Final Reward
1	76	1
2	81	1
3	87	1
4	83	1
5	76	1

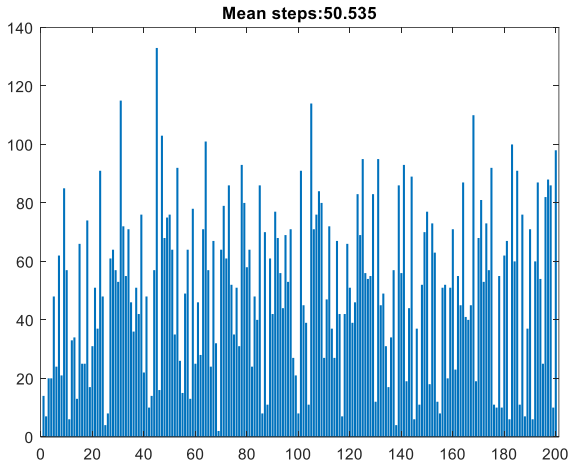
Gambaran koordinat akhir masing-masing objek setelah iterasi ke-lima ditunjukkan pada Gambar 4.26.



Gambar 4.26 koordinat akhir masing-masing objek pada skenario 3

Berdasarkan pada Tabel 4.10 dan Gambar 4.26, tampak bahwa *agent* berhasil menuju *target* setelah dilakukan lima kali iterasi. Perjalanan tercepat terjadi pada iterasi pertama dan ke-lima dengan jumlah *action* sebanyak 76.

Distribusi jumlah langkah untuk mencapai tujuan dalam setiap episode ditunjukkan pada Gambar 4.27.



Gambar 4.27 Distribusi jumlah langkah untuk mencapai tujuan pada skenario 3

Algoritma *Linear Function Approximation (LFA)* fokus pada meningkatkan kinerja agen dengan mengoptimalkan kebijakan agen. Dalam setiap iterasi, kebijakan agen diperbarui berdasarkan gradien kebijakan terhadap estimasi keuntungan yang diharapkan. Dalam pendekatan ini, fungsi linear digunakan untuk mendekati nilai kebijakan atau nilai fungsi di berbagai keadaan. Ini memungkinkan agen untuk belajar dari pengalaman dan menggeneralisasi ke keadaan yang belum pernah dilihat sebelumnya.

Penggunaan estimator kebijakan dan estimator nilai memungkinkan agen untuk memprediksi probabilitas tindakan berdasarkan keadaan dan memperkirakan nilai fungsi keadaan. Estimator ini terus diperbarui selama proses pembelajaran untuk mengikuti perubahan lingkungan. Dalam algoritma ini, agen terus-menerus berinteraksi dengan lingkungan dan memperbarui estimasi kebijakan dan nilai berdasarkan umpan balik yang diterima. Ini memungkinkan agen untuk mengadaptasi kebijakan berdasarkan perubahan dalam lingkungan.

Algoritma ini dapat diuji dengan melakukan serangkaian pengujian pada lingkungan. Distribusi jumlah langkah dan imbalan dalam pengujian dapat memberikan gambaran tentang kinerja agen dalam mencapai tujuan. Algoritma ini memiliki beberapa hyperparameter seperti tingkat pembelajaran *(alpha),* faktor diskon *(gamma)*, dan parameter lain yang perlu ditentukan. Memilih nilai *hyperparameter* yang sesuai dapat berdampak signifikan pada konvergensi dan kinerja algoritma.

Hasil pengujian dan pelatihan dapat membantu dalam mengukur efektivitas algoritma ini dalam mencapai tujuan dalam lingkungan tertentu. Dalam beberapa kasus, algoritma ini dapat memerlukan penyetelan yang cermat untuk mencapai hasil yang optimal. Dengan menggabungkan prinsip-prinsip kebijakan gradien dengan pendekatan *linear function approximation,* algoritma ini memberikan pendekatan yang kuat untuk melatih agen dalam mengambil keputusan yang optimal dalam lingkungan yang dinamis.

Pengujian yang dilakukan pada *single agent reinforcement learning* diimplementasikan untuk mengajari *agent* menghindari beberapa *obstacle* dan mencapai target. Beberapa algoritma diterapkan untuk melatih *agent* dalam pengambilan keputusan agar dapat berinteraksi dengan lingkungan *(environtment).* Algoritma yang diterapkan pada pengujian dengan beberapa obstacle dan lingkungan antara lain: *Monte Carlo Methods* dan *Linear Function Approximation.*

4.4. Hasil Pengujian *Monte Carlo Method*

Metode *Monte Carlo* digunakan untuk memperkirakan nilai-nilai optimal dan kebijakan optimal dalam *reinforcement learning* tanpa mengharuskan adanya informasi penuh tentang lingkungan dan tanpa menggunakan model lingkungan. Metode ini memperkirakan nilai-nilai optimal dengan menghitung rerata imbalan yang diperoleh dari jalur (trajectory) yang diambil agen dalam lingkungan. Metode *Monte Carlo* dibagi menjadi dua jenis yaitu *On-Policy Monte Carlo* dan *Off-Policy Monte Carlo.*

4.4.1 **On-Policy Monte Carlo**

On-policy monte carlo bertujuan untuk memperbarui dan meningkatkan kebijakan agen berdasarkan pengalaman nyata yang diperoleh dari interaksi dengan lingkungan menggunakan kebijakan yang sama. *On-Policy Monte Carlo* bekerja dengan menghasilkan episode-episode berdasarkan kebijakan yang digunakan oleh agen. Dalam setiap episode, agen berinteraksi dengan lingkungan dan mencatat tindakan yang diambil dan imbalan yang diterima. Episode-episode ini digunakan untuk memperkirakan nilai-nilai state dan state-action pair serta memperbarui kebijakan.

Langkah-langkah algoritma *On-Policy Monte Carlo*

1. Inisialisasi:
    a. Menginisialisasi lingkungan (`env`) menggunakan kelas `SAEnvironment`.

b. Variabel `isTraining` digunakan untuk menentukan apakah kode berada dalam mode pelatihan atau hanya menampilkan hasil yang sudah ada.

2. Inisialisasi Parameter:

a. Jika dalam mode pelatihan (`isTraining = true`), beberapa parameter penting diatur:

b. `NUM_ITERATIONS`: Jumlah iterasi (episode) yang akan dilakukan.

c. `gamma`: Faktor diskon untuk menghitung return.

d. `epsilon`: Probabilitas eksplorasi, yang mengontrol seberapa sering agen mengambil tindakan acak.

e. `min_epsilon`: Batas bawah untuk probabilitas eksplorasi.

f. `Q`: Matriks nilai tindakan (Q-values).

g. `Returns`: Matriks yang menyimpan jumlah return dan hitungan untuk setiap state-action pair.

h. `Policy`: Matriks kebijakan awal dengan probabilitas yang merata.

3. Loop Iterasi (Episode):

a. Loop utama yang akan berjalan sebanyak `NUM_ITERATIONS` kali.

b. Pada setiap iterasi (episode), lingkungan di-reset dan agen mulai berinteraksi dengan lingkungan.

4. Iterasi dalam Episode:

a. Agen melakukan interaksi dengan lingkungan selama episode, memilih tindakan berdasarkan kebijakan yang digunakan.

b. Imbalan yang diterima dan tindakan yang diambil dicatat dalam matriks `matrix_episode`.

5. Pembaruan dan Perhitungan Return:

   Setelah episode selesai, agen memeriksa setiap langkah yang diambil dalam episode dan menghitung return untuk setiap state-action pair yang belum pernah dikunjungi sebelumnya.

6. Perbaruan Nilai dan Kebijakan:

   Untuk setiap state-action pair yang belum pernah dikunjungi dalam episode, agen menghitung return dan memperbarui nilai tindakan (`Q`) serta kebijakan (`Policy`) berdasarkan return yang dihitung.

7. Penyimpanan dan Visualisasi:

   a. Jika dalam mode pelatihan, hasil dari iterasi disimpan dalam berbagai file (seperti `onpmc_q.mat`, `onpmc_returns.mat`, dll.).
   b. Visualisasi yang menunjukkan jumlah tindakan dan imbalan yang diperoleh dari setiap iterasi.

4.4.2 **Off-Policy Monte Carlo**

Tujuan utama dari *Off-Policy Monte Carlo* adalah memperbarui kebijakan dan memperkirakan nilai-nilai optimal berdasarkan pengalaman yang diperoleh dari kebijakan eksplorasi yang berbeda dengan kebijakan target yang ingin dipelajari. *Off-Policy Monte Carlo* memisahkan antara kebijakan eksplorasi *(epsilon-greedy)* dan kebijakan target yang ingin dipelajari. Pengalaman dikumpulkan dengan menjalankan kebijakan eksplorasi untuk mengumpulkan data dan kemudian menggunakannya untuk memperbarui kebijakan target.

Langkah-langkah algoritma *Off-Policy Monte Carlo*

1. Inisialisasi:
    a. Menginisialisasi lingkungan (`env`) menggunakan kelas `SAEnvironment`.
    b. Variabel `isTraining` digunakan untuk menentukan apakah kode berada dalam mode pelatihan atau hanya menampilkan hasil yang sudah ada.
2. Inisialisasi Parameter:
    a. Jika dalam mode pelatihan (`isTraining = true`), beberapa parameter penting diatur:
    b. `NUM_ITERATIONS`: Jumlah iterasi (episode) yang akan dilakukan.
    c. `gamma`: Faktor diskon untuk menghitung return.
    d. `epsilon`: Probabilitas eksplorasi untuk kebijakan eksplorasi.
    e. `min_epsilon`: Batas bawah untuk probabilitas eksplorasi.
    f. `Q`: Matriks nilai tindakan (Q-values).
    g. `C`: Matriks yang menyimpan kumulatif denominasi dari rumus importance sampling berbobot.
    h. `Policy`: Matriks kebijakan awal dengan probabilitas yang merata.
    i. Variabel lain untuk melacak iterasi dan imbalan.
3. Loop Iterasi (Episode):
    a. Loop utama yang akan berjalan sebanyak `NUM_ITERATIONS` kali.
    b. Pada setiap iterasi (episode), lingkungan di-reset dan agen mulai berinteraksi dengan lingkungan.
4. Iterasi dalam Episode:

Agen melakukan interaksi dengan lingkungan selama episode, memilih tindakan berdasarkan kebijakan eksplorasi dan kemudian kebijakan target.

5. Pembaruan dan Perhitungan Return:

Setelah episode selesai, agen memeriksa setiap langkah yang diambil dalam episode dan menghitung return untuk setiap langkah, menghitung G (return kumulatif) dan W (rasio importance sampling).

6. Perbaruan Nilai dan Kebijakan:

a. Untuk setiap langkah dalam episode, Q-value diperbarui berdasarkan rumus incremental update, di mana Q-value diperbarui dengan mempertimbangkan W dan return yang dihitung.
b. Kebijakan target diperbarui menjadi greedy berdasarkan Q-values yang diperbarui.

7. Penyimpanan dan Visualisasi:

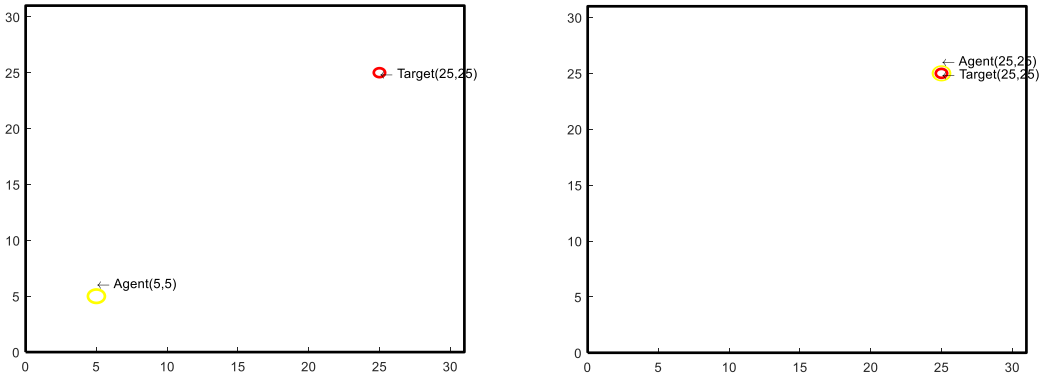
a. Jika dalam mode pelatihan, hasil dari iterasi disimpan dalam berbagai file (seperti `offpmc_q.mat`, `offpmc_c.mat`, dll.).
b. Visualisasi yang menunjukkan jumlah tindakan dan imbalan yang diperoleh dari setiap iterasi.

4.4.3. **Simulasi Tanpa Obstacle**

Pada simulasi ini dilakukan pergerakan agent menuju target tanpa adanya obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.28.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.28 Koordinat masing-masing objek simulasi tanpa obstacle pada lingkungan berbentuk persegi

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.11 dan Tabel 4.12 berikut ini.

Tabel 4.11 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk persegi On-Policy Monte Carlo

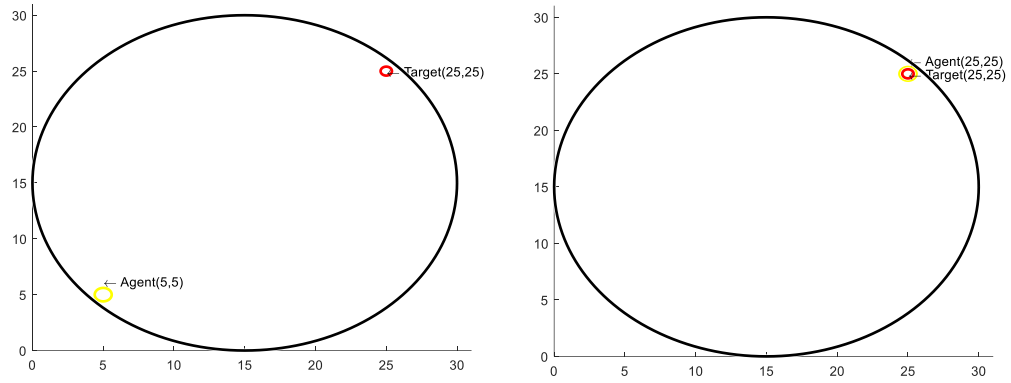
Iteration	Actions Taken	Final Reward
1	52	1
2	48	1
3	56	1
4	58	1
5	52	1

Tabel 4.12 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk persegi Off-Policy Monte Carlo

	Actions Taken	Final Reward
1	70	1
2	70	1
3	60	1
4	72	1
5	64	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.29.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.29 Koordinat masing-masing objek simulasi tanpa obstacle pada lingkungan berbentuk lingkaran

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.13 dan Tabel 4.14 berikut ini.

Tabel 4.13 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk lingkaran On-Policy Monte Carlo

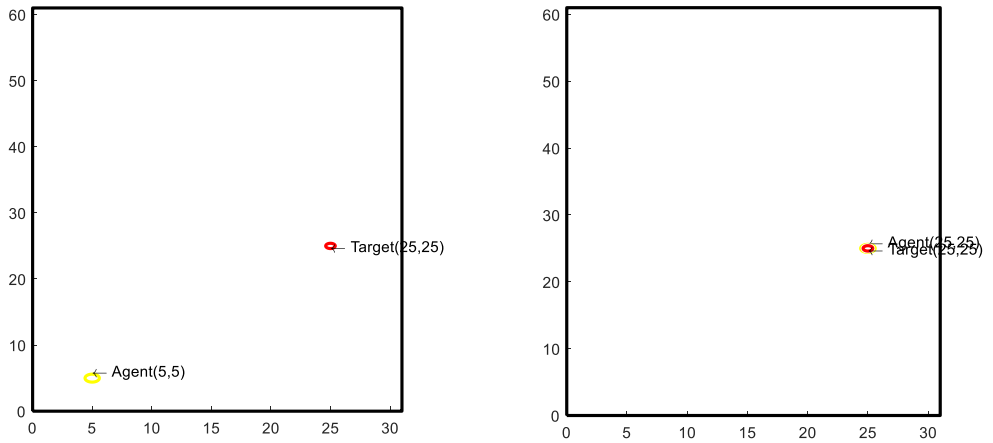
Iteration	Actions Taken	Final Reward
1	68	1
2	54	1
3	68	1
4	66	1
5	64	1

Tabel 4.14 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk lingkaran Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	78	1
2	80	1
3	56	1
4	84	1
5	94	1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi panjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.30.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.30 Koordinat masing-masing objek simulasi tanpa obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.15 dan Tabel 4.16 berikut ini.

Tabel 4.15 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk huruf On-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	76	1
2	74	1
3	98	1
4	112	1
5	58	1

Tabel 4.16 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk huruf Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	93	1
2	60	1
3	74	1
4	98	1
5	54	1

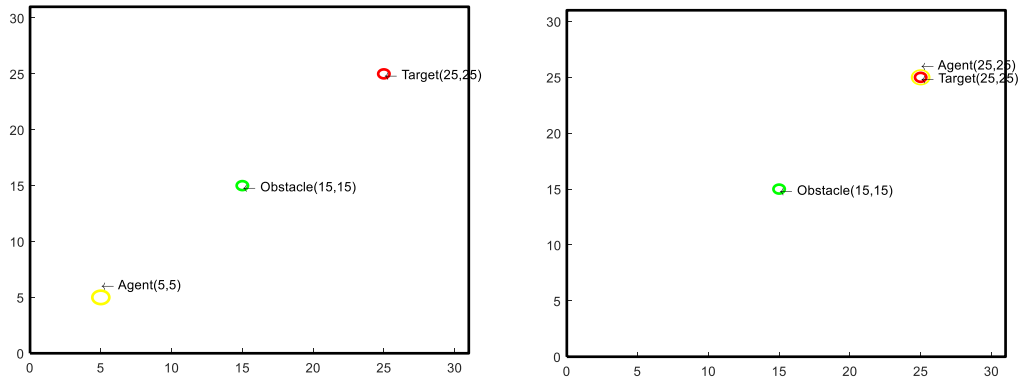
4.4.4. Simulasi Dengan 1 Obstacle

Pada simulasi ini dilakukan pergerakan agent menuju target dengan adanya satu obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh satu buah obstacle pada koordinat (15,15). Agent

bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.31.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.31 Koordinat masing-masing objek simulasi tanpa obstacle pada lingkungan berbentuk persegi

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.17 dan Tabel 4.18 berikut ini.

Tabel 4.17 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk persegi On-Policy Monte Carlo

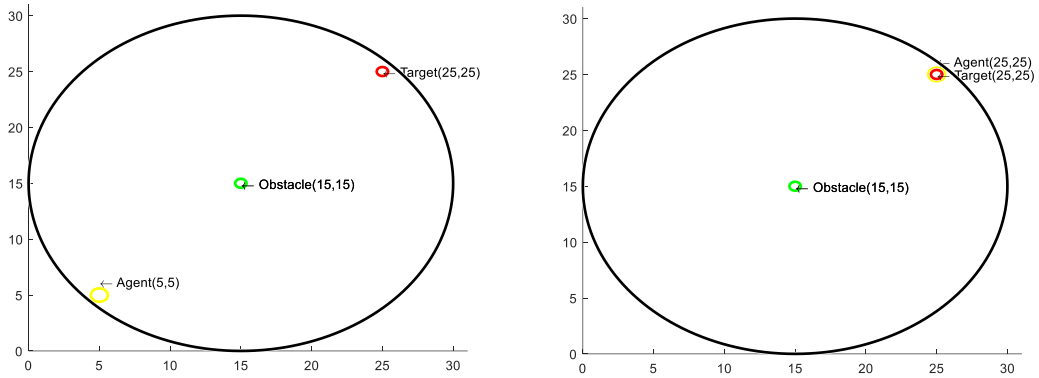
Iteration	Actions Taken	Final Reward
1	52	1
2	48	1
3	56	1
4	58	1
5	52	1

Tabel 4.18 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk persegi Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	70	1
2	70	1
3	60	1
4	72	1
5	64	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh satu buah obstacle pada koordinat (15,15). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.32.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.32 Koordinat masing-masing objek simulasi dengan 1 obstacle pada lingkungan berbentuk lingkaran

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.19 dan Tabel 4.20 berikut ini.

Tabel 4.19 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk lingkaran On-Policy Monte Carlo

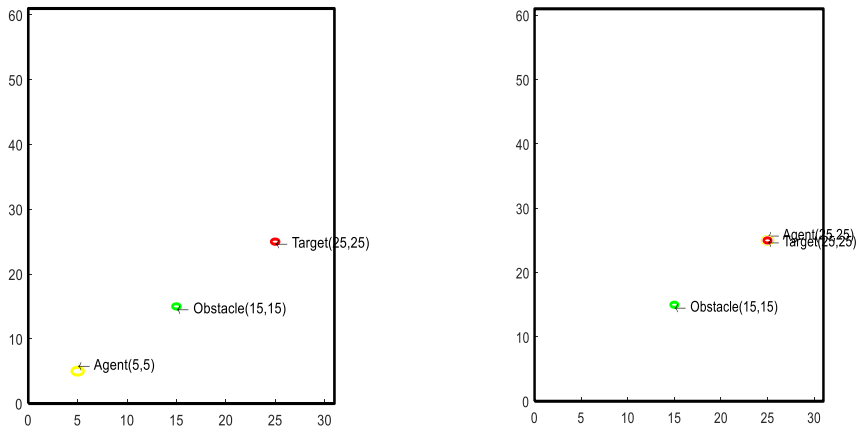
Iteration	Actions Taken	Final Reward
1	68	1
2	54	1
3	68	1
4	66	1
5	64	1

Tabel 4.20 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk lingkaran Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	78	1
2	80	1
3	56	1
4	84	1
5	94	1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi panjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh satu buah obstacle pada koordinat (15,15). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.33.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.33 Koordinat masing-masing objek simulasi dengan 1 obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.21 dan Tabel 4.22 berikut ini.

Tabel 4.21 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk huruf On-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	76	1
2	74	1
3	98	1
4	112	1
5	58	1

Tabel 4.22 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk huruf Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	93	1
2	60	1
3	74	1
4	98	1
5	54	1

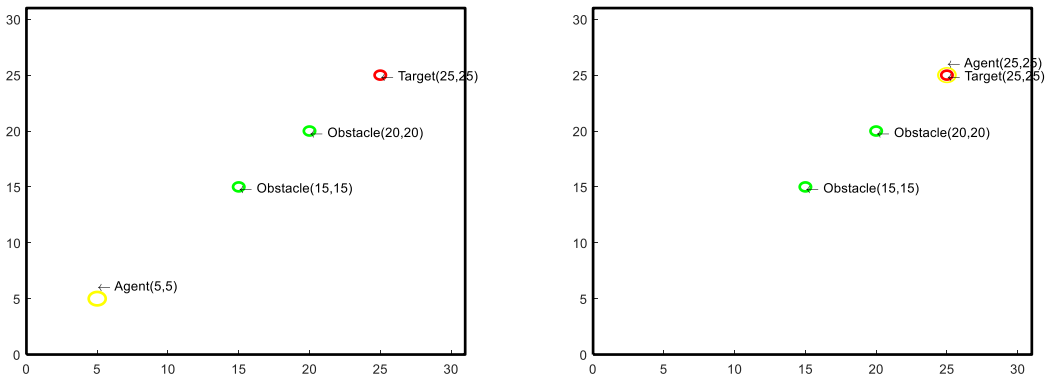
4.4.5. **Simulasi Dengan 2 Obstacle**

Pada simulasi ini dilakukan pergerakan agent menuju target dengan adanya dua obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (15,15) dan (20,20).

Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.34



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.34 Koordinat masing-masing objek simulasi dengan 2 obstacle pada lingkungan berbentuk persegi

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.23 dan Tabel 4.24 berikut ini.

Tabel 4.23 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk persegi On-Policy Monte Carlo

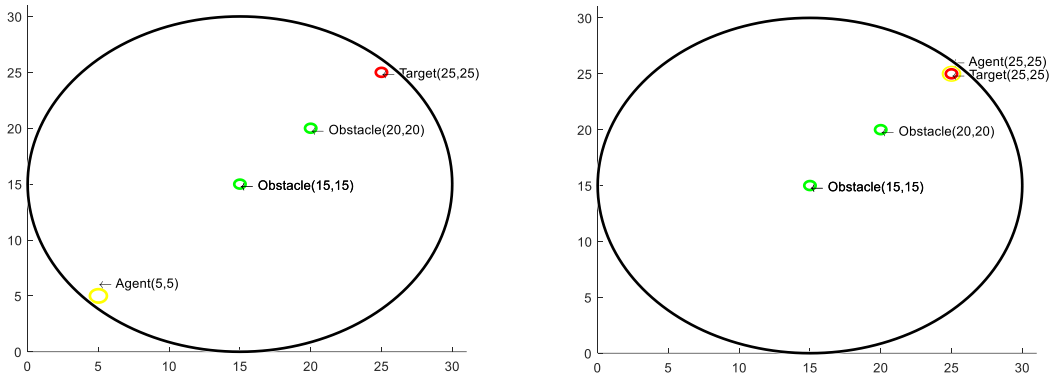
Iteration	Actions Taken	Final Reward
1	52	1
2	48	1
3	56	1
4	58	1
5	52	1

Tabel 4.24 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk persegi Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	70	1
2	70	1
3	48	-1
4	64	1
5	50	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (15,15) dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.35.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.35 Koordinat masing-masing objek simulasi dengan 2 obstacle pada lingkungan berbentuk lingkaran

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.25 dan Tabel 4.26 berikut ini.

Tabel 4.25 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk lingkaran On-Policy Monte Carlo

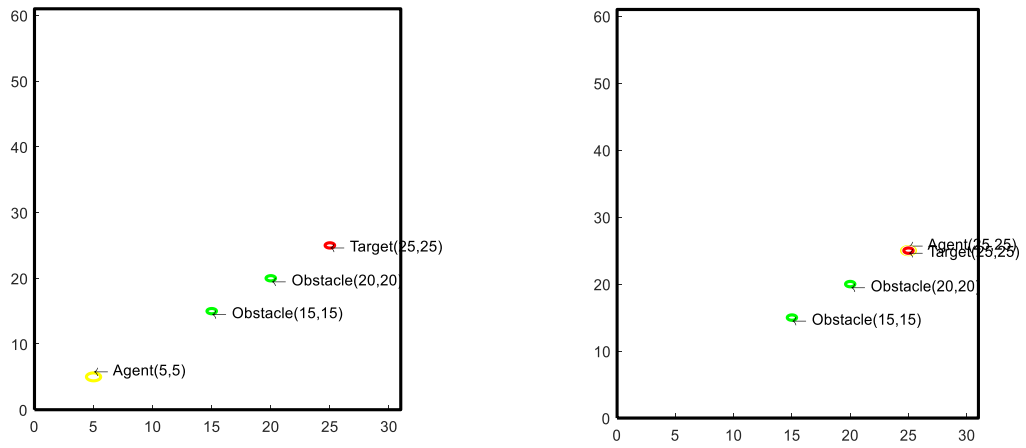
Iteration	Actions Taken	Final Reward
1	68	1
2	54	1
3	68	1
4	66	1
5	64	1

Tabel 4.26 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk lingkaran Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	54	-1
2	60	1
3	74	1
4	84	1
5	48	-1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi panjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (15,15) dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.36.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.36 Koordinat masing-masing objek simulasi dengan 2 obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.27 dan Tabel 4.28 berikut ini.

Tabel 4.27 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk huruf On-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	76	1
2	74	1
3	98	1
4	112	1
5	58	1

Tabel 4.28 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk huruf Off-Policy Monte Carlo

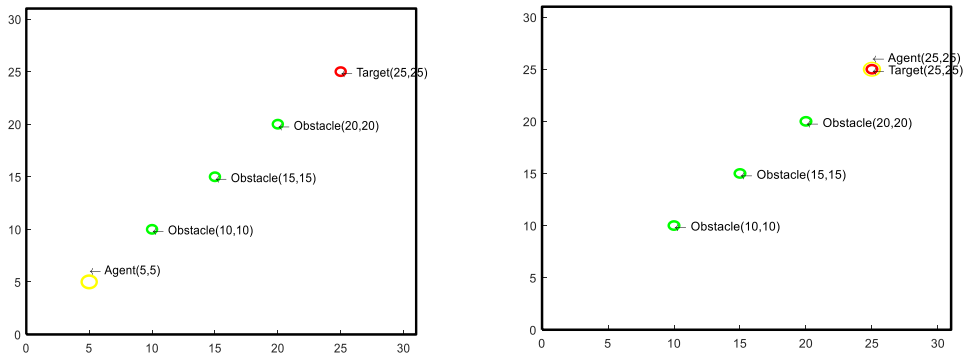
Iteration	Actions Taken	Final Reward
1	93	1
2	34	-1
3	74	-1
4	94	1
5	84	1

4.4.6. **Simulasi Dengan 3 Obstacle**

Pada simulasi ini dilakukan pergerakan agent menuju target dengan adanya tiga obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (10,10), (15,15), dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.37.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.37 Koordinat masing-masing objek simulasi dengan 3 obstacle pada lingkungan berbentuk persegi

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.29 dan Tabel 4.30 berikut ini.

Tabel 4.29 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk persegi On-Policy Monte Carlo

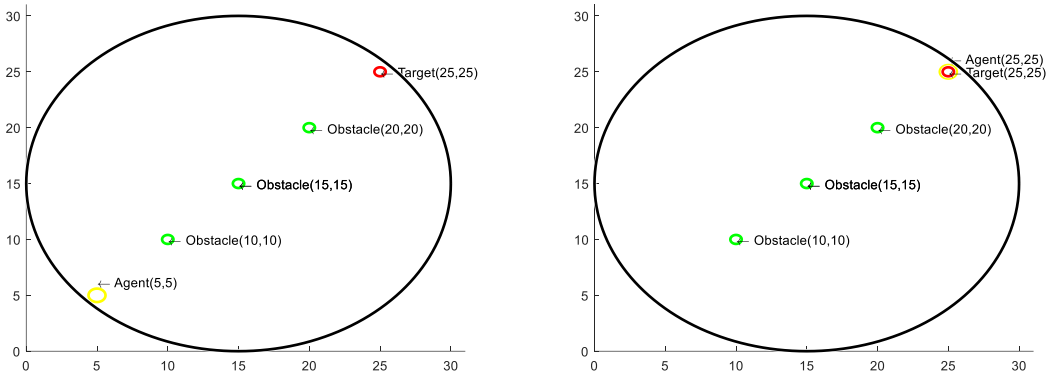
Iteration	Actions Taken	Final Reward
1	14	-1
2	60	1
3	16	-1
4	46	1
5	62	1

Tabel 4.30 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk persegi Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	70	1
2	70	1
3	60	1
4	72	1
5	64	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh tiga buah obstacle pada koordinat (10,10), (15,15), dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.38.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.38 Koordinat masing-masing objek simulasi dengan 3 obstacle pada lingkungan berbentuk lingkaran

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.31 dan Tabel 4.32 berikut ini.

Tabel 4.31 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk lingkaran On-Policy Monte Carlo

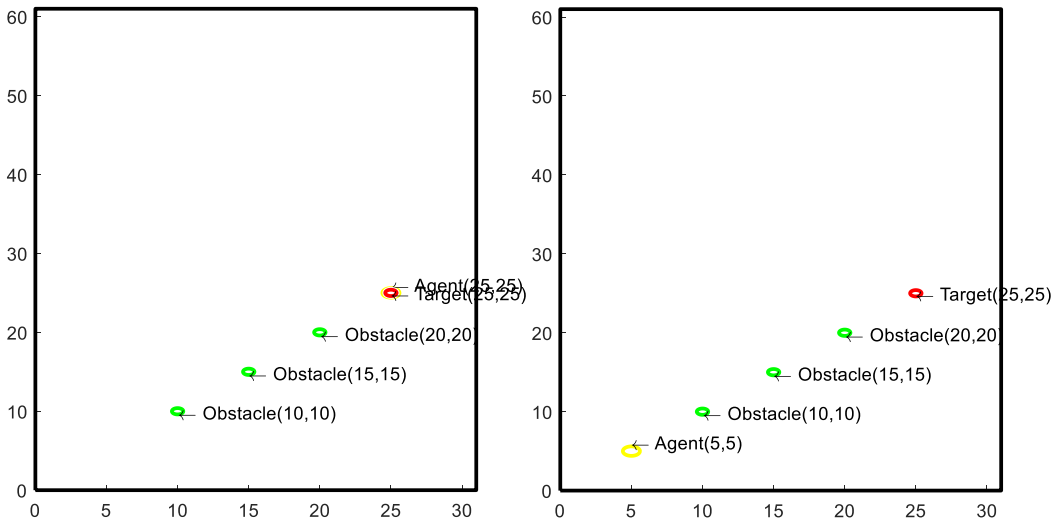
Iteration	Actions Taken	Final Reward
1	16	-1
2	66	1
3	64	1
4	58	1
5	62	1

Tabel 4.32 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk lingkaran Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	78	1
2	80	1
3	56	1
4	84	1
5	94	1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi panjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh tiga buah obstacle pada koordinat (10,10), (15,15), dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.39.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.39 Koordinat masing-masing objek simulasi dengan 3 obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.33 dan Tabel 4.34 berikut ini.

Tabel 4.33 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk huruf On-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	16	-1
2	85	1
3	14	-1
4	76	1
5	88	1

Tabel 4.34 umlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk huruf Off-Policy Monte Carlo

Iteration	Actions Taken	Final Reward
1	93	1
2	60	1
3	74	1
4	98	1
5	54	1

Berdasarkan hasil penelitian mengenai metode Monte Carlo dalam *single agent reinforcement learning*, terdapat beberapa poin utama:

1. Metode Monte Carlo merupakan alat yang kuat dalam reinforcement learning, terutama ketika agen harus belajar dari pengalaman yang dihasilkan dari episode lengkap. Pendekatan ini berguna dalam situasi di mana tidak ada pengetahuan sebelumnya tentang lingkungan yang diperlukan untuk membuat keputusan.
2. Hasil eksperimen menunjukkan bahwa metode Monte Carlo mampu mengajari agen untuk beroperasi dengan baik dalam lingkungan dengan berbagai bentuk seperti persegi, lingkaran, dan persegi panjang. Ini mengindikasikan kemampuan adaptasi metode Monte Carlo terhadap variasi lingkungan yang luas.

3. Jumlah obstacle dalam lingkungan dapat memengaruhi performa agen dalam belajar kebijakan yang optimal. Semakin banyak obstacle, semakin sulit bagi agen untuk mencapai tujuan, dan ini mengakibatkan waktu yang lebih lama untuk konvergensi.

4. Kesulitan tugas reinforcement learning sangat tergantung pada jumlah obstacle dan sebaran mereka. Kombinasi dari jumlah obstacle dan bentuk lingkungan memberikan tingkat kesulitan yang berbeda bagi agen dalam menemukan solusi yang optimal.

5. Penggunaan metode Monte Carlo dengan bijak memerlukan eksplorasi yang cermat dan pengontrolan parameter yang tepat sesuai dengan tugas dan lingkungan tertentu.

Kesimpulannya, metode Monte Carlo adalah metode yang handal dalam reinforcement learning yang mampu mengatasi variasi dalam lingkungan dan jumlah obstacle. Namun, performa agen sangat dipengaruhi oleh kompleksitas lingkungan dan sebaran obstacle.

4.5 Hasil Pengujian Linear Function Approximation

Linear Function Approximation (LFA) bertujuan untuk mengatasi masalah dimensi tinggi dari keadaan (state) dan tindakan (action) dalam lingkungan yang kompleks. Dalam konteks reinforcement learning untuk single-agent, tujuan utama LFA adalah untuk mengaproksimasi fungsi nilai (seperti Q-value atau nilai state) dengan menggunakan kombinasi linear dari fitur-fitur yang diambil dari keadaan dan tindakan.

Langkah-langkah Algoritma *Linear Function Approximation*

1. Inisialisasi Lingkungan:

   env = SAEnvironment;: Objek lingkungan dibuat menggunakan kelas SAEnvironment.

2. Pengaturan Parameter:

   a. alpha = 0.1;: Tingkat pembelajaran (learning rate) untuk algoritma LFA.
   b. gamma = 0.9;: Faktor diskon untuk menilai pengaruh imbalan di masa depan.
   c. maxItr = 3000;: Jumlah maksimum iterasi untuk mengakhiri satu episode.
   d. estimator = LFAEstimator(env,alpha);: Membuat objek estimator LFA dengan parameter lingkungan dan tingkat pembelajaran.

3. Inisialisasi Mode Training:

   a. Jika isTraining bernilai true, variabel berikut diinisialisasi:
      1. NUM_ITERATIONS: Jumlah iterasi pelatihan.
      2. epsilon: Nilai ε dalam kebijakan ε-greedy.
      3. min_epsilon: Nilai minimum untuk ε.
      4. iterationCount(NUM_ITERATIONS): Array untuk menyimpan jumlah langkah dalam setiap iterasi.
      5. rwd(NUM_ITERATIONS): Array untuk menyimpan hadiah dalam setiap iterasi.

4. Inisialisasi Mode Pengujian:

   b. Jika isTraining dan isTesting bernilai false, variabel berikut diinisialisasi:
      1. NUM_ITERATIONS: Jumlah iterasi pengujian.

2. iterationCount(NUM_ITERATIONS): Array untuk menyimpan jumlah langkah dalam setiap iterasi pengujian.
3. rwd(NUM_ITERATIONS): Array untuk menyimpan hadiah dalam setiap iterasi pengujian.
4. Memuat bobot LFA yang telah dilatih sebelumnya dari file 'onp_lfa_weights.mat'.
5. Menetapkan bobot tersebut pada objek estimator dengan estimator.set_weights(Weights).

5. Loop Utama:
   a. Loop ini akan dilakukan sebanyak NUM_ITERATIONS kali.
   b. Lingkungan direset ke posisi awal.
   c. Jika bukan mode pelatihan atau pengujian, lingkungan ditampilkan menggunakan env.render().
6. Eksplorasi dan Eksplotasi:
   a. Mengambil tindakan awal menggunakan kebijakan ε-greedy.
   b. Memperkirakan nilai dari tindakan tersebut menggunakan estimator LFA.
7. Loop Episode:
   a. Loop ini berjalan selama episode belum selesai (done = false).
   b. Dalam loop episode, agen berinteraksi dengan lingkungan:

      Jika batas maksimum iterasi tercapai, episode dihentikan.

      1. Mengambil tindakan menggunakan kebijakan ε-greedy.
      2. Melakukan tindakan dan mendapatkan hasil berikutnya dari lingkungan.

3. Jika bukan mode pelatihan, lingkungan ditampilkan.
4. Menghitung dan memperbarui nilai Q menggunakan algoritma On-Policy LFA.

8. Penyimpanan dan Visualisasi:

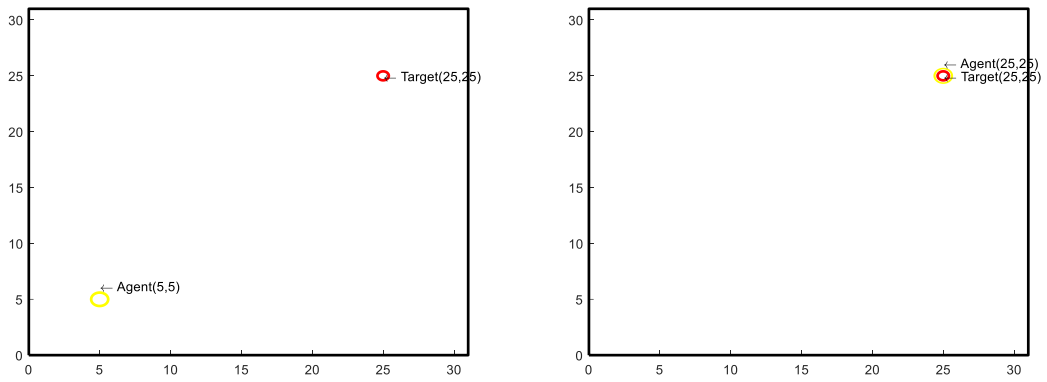
a. Jika dalam mode pelatihan, bobot estimator LFA disimpan dalam file 'onp_lfa_weights.mat'.
b. Data jumlah langkah dan hadiah disimpan dalam file terpisah.
c. Grafik batang jumlah langkah dan hadiah ditampilkan jika dalam mode pelatihan atau pengujian.

4.5.1. **Simulasi Tanpa Obstacle**

Pada simulasi ini dilakukan pergerakan agent menuju target tanpa adanya obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.40.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.40 Koordinat masing-masing objek simulasi tanpa obstacle pada lingkungan berbentuk persegi

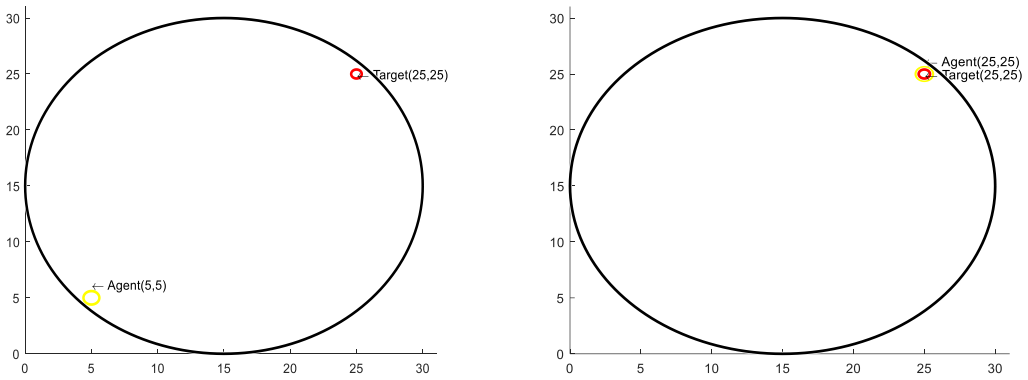
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.35 berikut ini.

Tabel 4.35 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk persegi Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	82	1
2	72	1
3	73	1
4	70	1
5	72	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.41.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.41 Koordinat masing-masing objek simulasi tanpa obstacle pada lingkungan berbentuk lingkaran

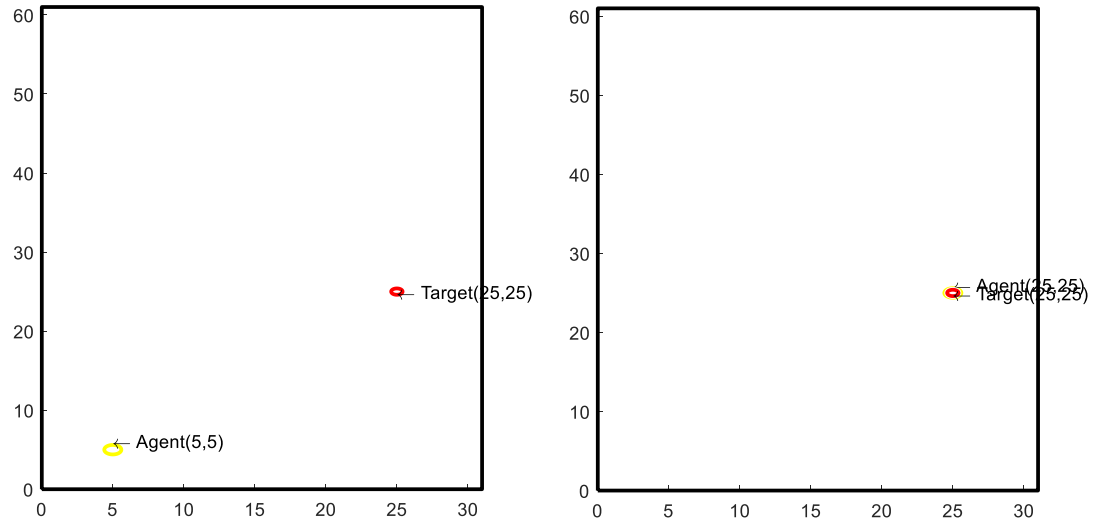
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.36 berikut ini.

Tabel 4.36 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk lingkaran Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	64	1
2	50	1
3	80	1
4	68	1
5	94	1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi 87anjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.42.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.42 Koordinat masing-masing objek simulasi tanpa obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.37 berikut ini.

Tabel 4.37 Jumlah action dan reward simulasi tanpa obstacle pada lingkungan berbentuk huruf Linear Function Approximation

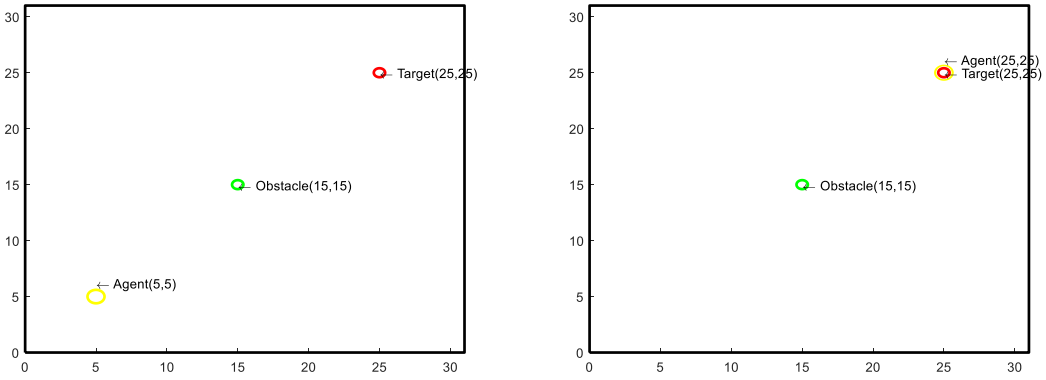
Iteration	Actions Taken	Final Reward
1	96	1
2	66	1
3	104	1
4	82	1
5	80	1

4.5.2. **Simulasi Dengan 1 Obstacle**

Pada simulasi ini dilakukan pergerakan agent menuju target dengan adanya satu obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh satu buah obstacle pada koordinat (15,15). Agent bergerak menuju target tanpa adanya obstacle. Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.43.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.43 Koordinat masing-masing objek simulasi dengan 1 obstacle pada lingkungan berbentuk persegi

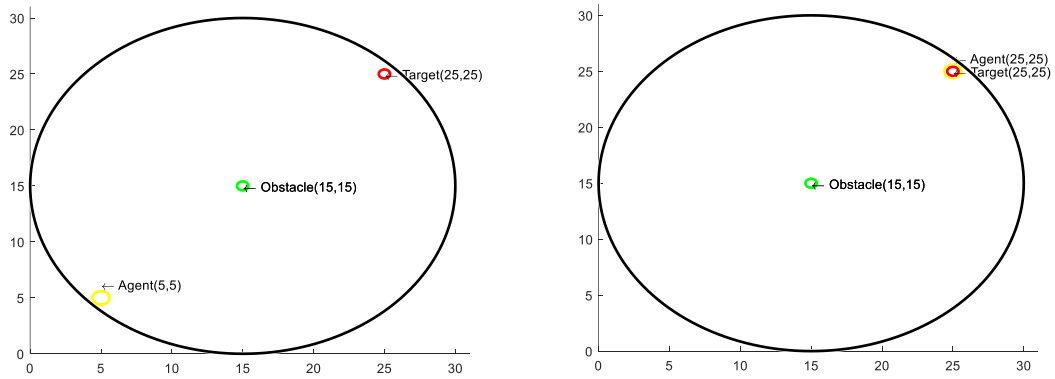
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.38 berikut ini.

Tabel 4.38 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk persegi Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	73	1
2	66	1
3	87	1
4	84	1
5	93	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh satu buah obstacle pada koordinat (15,15). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.44.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.44 Koordinat masing-masing objek simulasi dengan 1 obstacle pada lingkungan berbentuk lingkaran

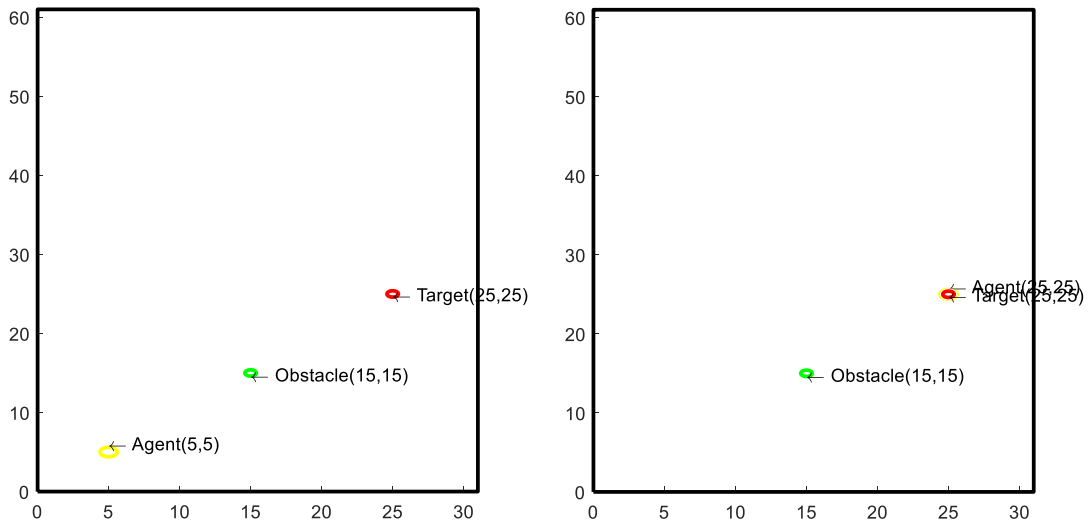
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.39 berikut ini.

Tabel 4.39 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk lingkaran Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	185	1
2	197	1
3	196	1
4	113	1
5	115	1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi panjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh satu buah obstacle pada koordinat (15,15). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.45.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.45 Koordinat masing-masing objek simulasi dengan 1 obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.40 berikut ini.

Tabel 4.40 Jumlah action dan reward simulasi dengan 1 obstacle pada lingkungan berbentuk huruf Linear Function Approximation

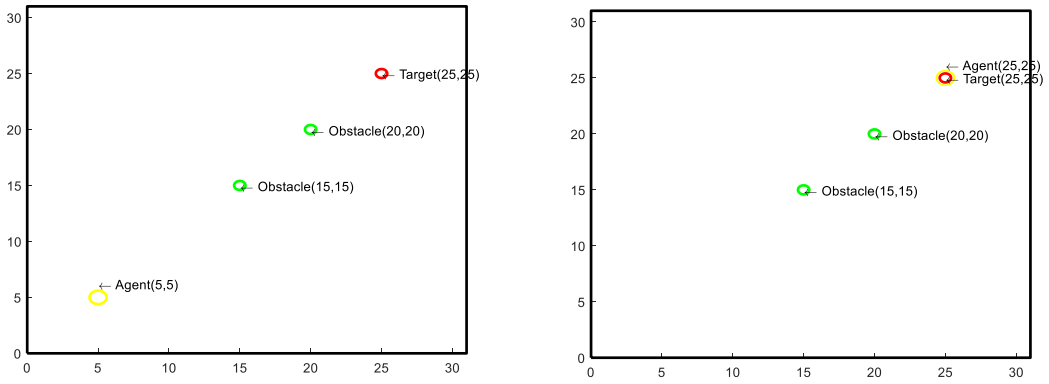
Iteration	Actions Taken	Final Reward
1	106	1
2	120	1
3	123	1
4	139	1
5	146	1

4.5.3. **Simulasi Dengan 2 Obstacle**

Pada simulasi ini dilakukan pergerakan agent menuju target dengan adanya dua obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (15,15) dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.46.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.46 Koordinat masing-masing objek simulasi dengan 2 obstacle pada lingkungan berbentuk persegi

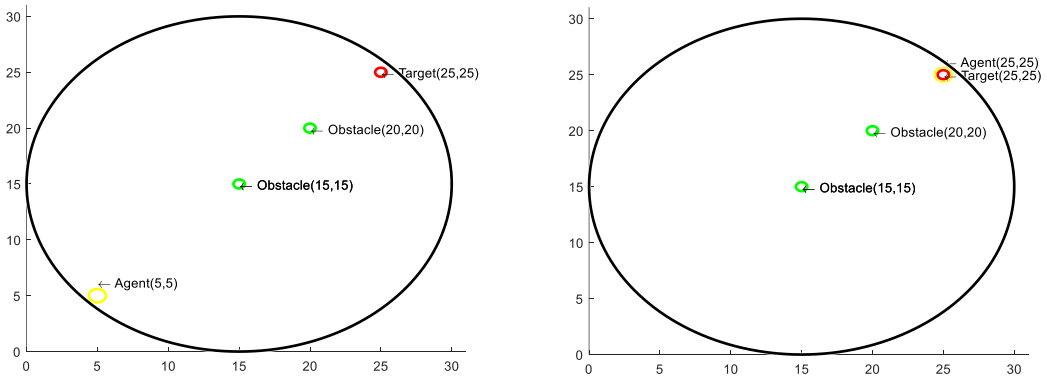
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.41 berikut ini.

Tabel 4.41 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk persegi Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	76	1
2	81	1
3	87	1
4	83	1
5	76	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (15,15) dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.47.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.47 Koordinat masing-masing objek simulasi dengan 2 obstacle pada lingkungan berbentuk lingkaran

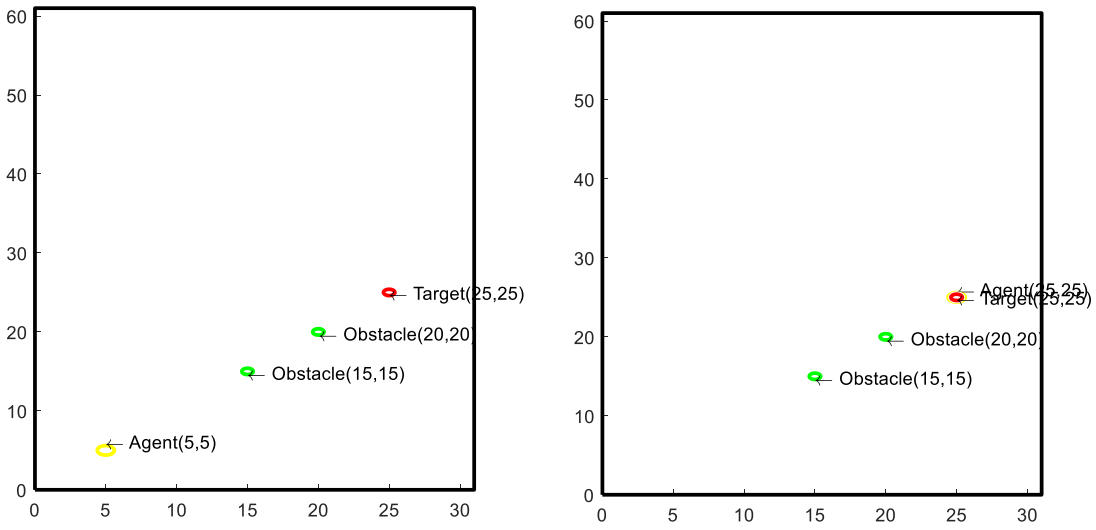
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.42 berikut ini.

Tabel 4.42 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk lingkaran Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	118	1
2	103	1
3	117	1
4	105	1
5	100	1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi panjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (15,15) dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.48.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.48 Koordinat masing-masing objek simulasi dengan 2 obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.43 berikut ini.

Tabel 4.43 Jumlah action dan reward simulasi dengan 2 obstacle pada lingkungan berbentuk huruf Linear Function Approximation

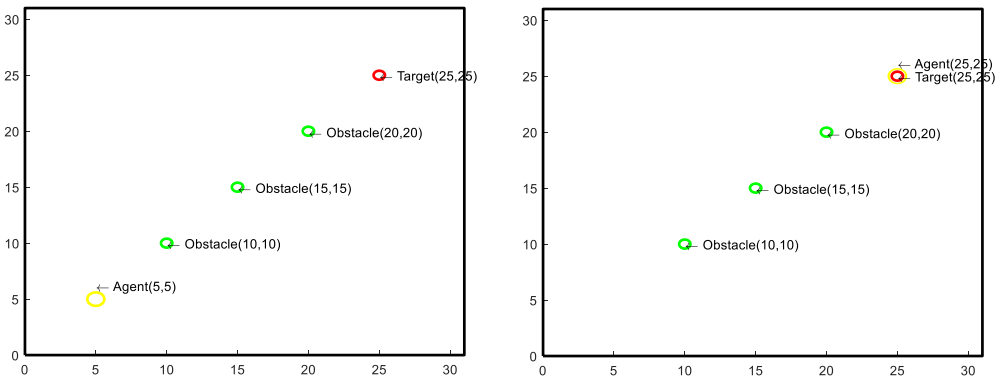
Iteration	Actions Taken	Final Reward
1	118	1
2	118	1
3	114	1
4	138	1
5	168	1

4.5.4. **Simulasi Dengan 3 Obstacle**

Pada simulasi ini dilakukan pergerakan agent menuju target dengan adanya tiga obstacle. Simulasi dilakukan dengan tiga bentuk lingkungan yang berbeda yaitu persegi, lingkaran, dan huruf.

A Lingkungan Berbentuk Persegi

Lingkungan berbentuk persegi berukuran 30x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh dua buah obstacle pada koordinat (10,10), (15,15), dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.49.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.49 Koordinat masing-masing objek simulasi dengan 3 obstacle pada lingkungan berbentuk persegi

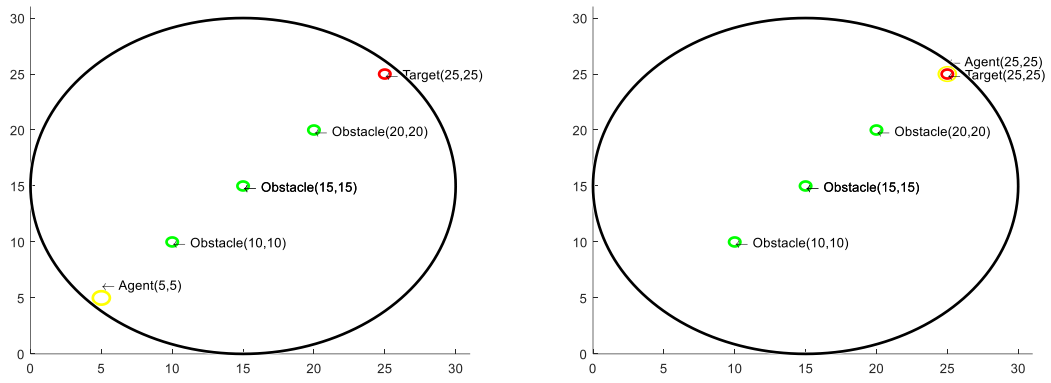
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.44 berikut ini.

Tabel 4.44 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk persegi Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	82	1
2	72	1
3	83	1
4	85	1
5	81	1

B Lingkungan Berbentuk Lingkaran

Lingkungan berbentuk lingkaran dengan diameter 30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh tiga buah obstacle pada koordinat (10,10), (15,15), dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.50.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.50 Koordinat masing-masing objek simulasi dengan 3 obstacle pada lingkungan berbentuk lingkaran

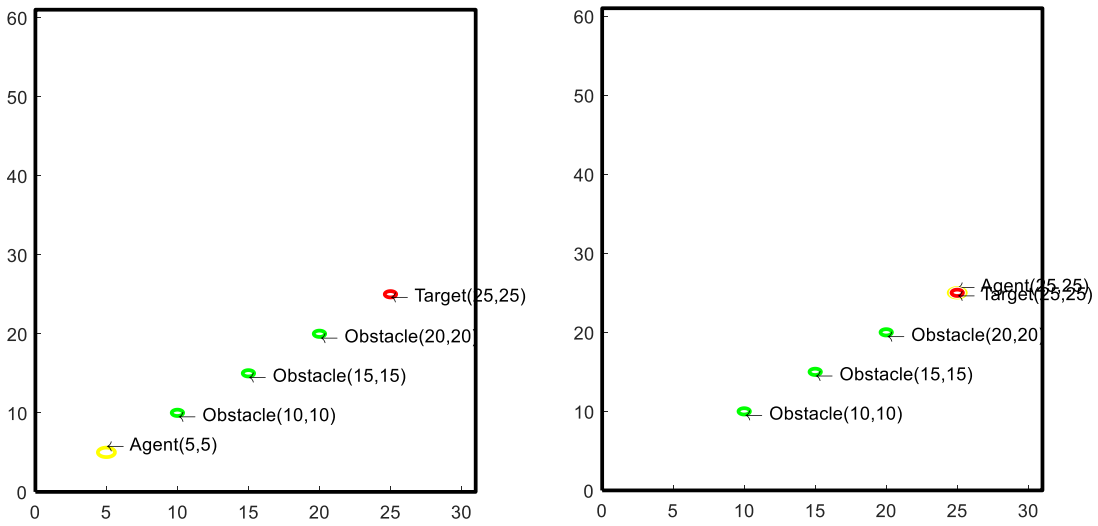
Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.45 berikut ini.

Tabel 4.45 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk lingkaran Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	96	1
2	82	1
3	66	1
4	91	1
5	84	1

C Lingkungan Berbentuk Huruf

Lingkungan berbentuk huruf I atau persegi panjang berukuran 60x30, dengan koordinat awal agent adalah (5,5) dan koordinat target adalah (25,25). Agent bergerak menuju target dengan dihalangi oleh tiga buah obstacle pada koordinat (10,10), (15,15), dan (20,20). Gambaran koordinat awal dan akhir masing-masing objek ditunjukkan pada Gambar 4.51.



(a) Koordinat Awal (b) Koordinat Akhir

Gambar 4.51 Koordinat masing-masing objek simulasi dengan 3 obstacle pada lingkungan berbentuk huruf

Iterasi dilakukan sebanyak 5 kali dengan jumlah *action* dan *reward* ditunjukkan pada Tabel 4.46 berikut ini.

Tabel 4.46 Jumlah action dan reward simulasi dengan 3 obstacle pada lingkungan berbentuk huruf Linear Function Approximation

Iteration	Actions Taken	Final Reward
1	96	1
2	82	1
3	110	1
4	90	1
5	101	1

Berdasarkan hasil penelitian mengenai Linear Function Approximation dalam single agent reinforcement learning, dapat disimpulkan beberapa poin utama:

1. Linear Function Approximation (LFA) adalah metode yang efektif untuk mengatasi masalah ruang parameter yang besar dalam reinforcement learning.

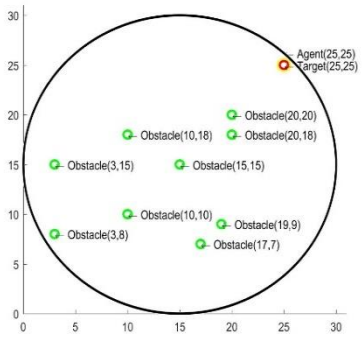
Dengan menggunakan aproksimasi fungsi linear, penghitungan parameter kebijakan menjadi lebih efisien.

2. Hasil eksperimen menunjukkan bahwa LFA dapat berhasil mengajari agen untuk beroperasi dalam lingkungan yang beragam dengan tiga bentuk berbeda. Ini menunjukkan fleksibilitas LFA dalam menangani variasi dalam lingkungan.
3. Jumlah obstacle dalam lingkungan mempengaruhi tingkat kesulitan tugas reinforcement learning. Semakin banyak obstacle, semakin sulit bagi agen untuk mempelajari kebijakan yang optimal. Oleh karena itu, pengaturan jumlah obstacle perlu diperhatikan dalam desain lingkungan.
4. Efisiensi dan keberhasilan LFA dalam melatih agen bergantung pada sejumlah faktor, termasuk jumlah obstacle, ukuran lingkungan, dan sebaran obstacle. Ini menunjukkan bahwa penelitian lebih lanjut dapat dilakukan untuk mengoptimalkan penggunaan LFA dalam konteks yang lebih kompleks.

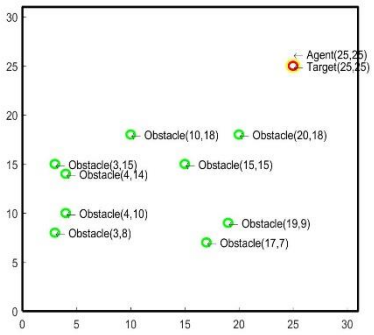
Dalam keseluruhan, hasil penelitian menunjukkan bahwa Linear Function Approximation adalah alat yang bermanfaat dalam memungkinkan agen untuk belajar dan beroperasi dalam lingkungan yang beragam dan kompleks.

4.6 Pembahasan Monte Carlo dan Linear Function Approximation

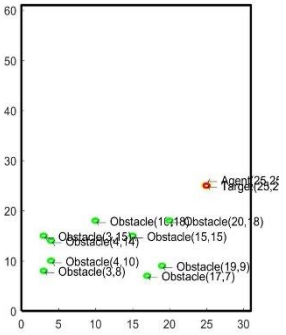
Dari hasil yang didapatkan di atas saya akan membandingkan dengan melakukan 50 iterasi dengan penyebaran obstacle di dalam 3 lingkungan yang dibuat seperti pada gambar 4.52, 4.53, 4.54 berikut



Gambar 4.53 Lingkungan Lingkaran



Gambar 4.52 Lingkungan Persegi



Gambar 4.54 Lingkungan Huruf

Sehingga didapatkan jumlah tindakan dari 50 iterasi tersebut beserta waktu yang telah di lalui oleh agent selama bergerak di dalam lingkungan.

4.6.1 **Monte Carlo Lingkungan Persegi**

50 kali iterasi dilakukan, sehingga hasil yang didapatkan pada table 4.47.

Tabel 4.47 Hasil Jumlah Tindakan dan Reward dari Agent

Iterasi	Jumlah Tindakan	Reward
1	122	1
2	60	1

3	52	1
4	54	1
5	52	1
6	58	1
7	66	1
8	62	1
9	70	1
10	68	1
11	44	1
12	48	1
13	60	1
14	52	1
15	54	1
16	66	1
17	74	1
18	58	1
19	68	1
20	60	1
21	58	1
22	48	1
23	60	1
24	50	1
25	62	1
26	60	1
27	62	1
28	76	1
29	70	1
30	54	1
31	54	1
32	76	1

33	68	1
34	72	1
35	190	1
36	54	1
37	56	1
38	62	1
39	48	1
40	50	1
41	58	1
42	52	1
43	66	1
44	96	1
45	68	1
46	50	1
47	60	1
48	80	1
49	86	1
50	46	1

Dari 50 iterasi tersebut dilakukan selama 357.68 detik / 5.9613 menit dengan jumlah tindakan paling banyak 190 tindakan serta tindakan paling sedikit 44 tindakan dengan rata-rata tindakan 64,80 dan dari 50 iterasi tersebut tidak ada yang menabrak obstacle.

4.6.2 **Monte Carlo Lingkungan Lingkaran**

50 kali iterasi dilakukan, sehingga hasil yang didapatkan pada table 4.48.

Tabel 4.48 Hasil Jumlah Tindakan dan Reward dari Agent

Iterasi	Jumlah Tindakan	Reward
1	78	1
2	80	1
3	56	1
4	84	1
5	94	1
6	44	1
7	66	1
8	32	-1
9	68	1
10	68	1
11	60	1
12	114	1
13	54	1
14	68	1
15	74	1
16	54	1
17	82	1
18	50	-1
19	62	1
20	72	1
21	66	1
22	52	1
23	54	1
24	82	-1
25	62	1
26	60	1
27	32	-1
28	56	1

29	60	1
30	100	1
31	96	1
32	80	1
33	62	1
34	20	-1
35	74	1
36	50	1
37	66	1
38	80	1
39	78	1
40	68	1
41	56	1
42	60	1
43	82	1
44	28	-1
45	64	1
46	64	1
47	70	1
48	72	1
49	82	1
50	72	1

Dari 50 iterasi tersebut dilakukan selama 365.91detik / 6.0985 menit dengan jumlah tindakan paling banyak 114 tindakan serta tindakan paling sedikit 20 tindakan dengan rata-rata tindakan 66.16 dan dari 50 iterasi tersebut terdapat 6 yang menabrak obstacle.

4.6.2 **Monte Carlo Lingkungan Huruf**

50 kali iterasi dilakukan, sehingga hasil yang didapatkan pada table 4.49.

Tabel 4.49 Hasil Jumlah Tindakan dan Reward dari Agent

Iterasi	Jumlah Tindakan	Reward
1	88	1
2	66	1
3	60	1
4	70	1
5	65	1
6	106	1
7	76	1
8	72	1
9	54	1
10	46	1
11	92	1
12	56	1
13	48	1
14	46	1
15	48	1
16	52	1
17	64	1
18	52	1
19	64	1
20	70	1
21	72	1
22	56	1
23	58	1
24	88	1
25	72	1

26	56	1
27	76	1
28	54	1
29	54	1
30	68	1
31	56	1
32	64	1
33	72	1
34	54	1
35	62	1
36	58	1
37	124	1
38	68	1
39	52	1
40	48	1
41	60	1
42	58	1
43	52	1
44	80	1
45	70	1
46	62	1
47	136	1
48	78	1
49	58	1
50	116	1

Dari 50 iterasi tersebut dilakukan selama 372.74 detik / 6.2123 menit dengan jumlah tindakan paling banyak 136 tindakan serta tindakan paling sedikit 46 tindakan dengan rata-rata tindakan 67.54 dan dari 50 iterasi tersebut tidak ada yang menabrak obstacle.

4.6.3 **Linear Function Approximation Lingkungan Persegi**

50 kali iterasi dilakukan, sehingga hasil yang didapatkan pada table 4.50.

Tabel 4. 50 Hasil Jumlah Tindakan dan Reward dari Agent

Iterasi	Jumlah Tindakan	Reward
1	104	1
2	66	1
3	66	1
4	74	1
5	67	1
6	66	1
7	54	1
8	116	1
9	60	1
10	52	1
11	64	1
12	62	1
13	66	1
14	82	1
15	70	1
16	68	1
17	56	1
18	50	1
19	50	1
20	68	1
21	74	1
22	46	1

23	56	1
24	56	1
25	90	1
26	66	1
27	70	1
28	82	1
29	76	1
30	72	1
31	82	1
32	62	1
33	90	1
34	70	1
35	106	1
36	80	1
37	56	1
38	74	1
39	58	1
40	58	1
41	56	1
42	58	1
43	72	1
44	55	1
45	85	1
46	88	1
47	80	1
48	58	1
49	56	1
50	82	1

Dari 50 iterasi tersebut dilakukan selama 225.23 detik / 3.7538 menit dengan jumlah tindakan paling banyak 116 tindakan serta tindakan paling sedikit 46 tindakan dengan rata-rata tindakan 69.5 dan dari 50 iterasi tersebut tidak ada yang menabrak obstacle.

4.6.4 **Linear Function Approximation Lingkungan Lingkaran**

50 kali iterasi dilakukan, sehingga hasil yang didapatkan pada table 4.51.

Tabel 4.51 Hasil Jumlah Tindakan dan Reward dari Agent

Iterasi	Jumlah Tindakan	Reward
1	66	1
2	68	1
3	82	1
4	84	1
5	74	1
6	60	1
7	54	1
8	62	1
9	46	-1
10	28	-1
11	56	1
12	64	1
13	68	1
14	32	-1
15	58	-1
16	72	1
17	80	1
18	64	1
19	64	1

20	82	1
21	72	1
22	54	1
23	64	1
24	40	-1
25	66	1
26	90	1
27	90	1
28	18	-1
29	50	1
30	56	1
31	58	1
32	72	1
33	74	1
34	60	1
35	66	1
36	76	1
37	34	-1
38	74	1
39	22	-1
40	24	-1
41	76	1
42	82	1
43	48	1
44	30	-1
45	74	1
46	78	1
47	86	1
48	18	-1
49	66	1

50	64	1

Dari 50 iterasi tersebut dilakukan selama 380.32 detik / 6.3372 menit dengan jumlah tindakan paling banyak 116 tindakan serta tindakan paling sedikit 46 tindakan dengan rata-rata tindakan 60.92 dan dari 50 iterasi tersebut terdapat 11 yang menabrak obstacle.

4.6.5 **Linear Function Approximation Lingkungan Huruf**

50 kali iterasi dilakukan, sehingga hasil yang didapatkan pada table 4.52.

Tabel 4.52 Hasil Jumlah Tindakan dan Reward dari Agent

Iterasi	Jumlah Tindakan	Reward
1	64	1
2	78	1
3	60	1
4	60	1
5	86	1
6	64	1
7	84	1
8	92	1
9	60	1
10	60	1
11	50	1
12	62	1
13	62	1
14	70	1
15	52	1
16	72	1
17	120	1

18	72	1
19	86	1
20	50	1
21	54	1
22	166	1
23	50	1
24	60	1
25	52	1
26	64	1
27	95	1
28	70	1
29	56	1
30	74	1
31	70	1
32	56	1
33	50	1
34	68	1
35	62	1
36	90	1
37	114	1
38	48	1
39	96	1
40	68	1
41	72	1
42	114	1
43	60	1
44	72	1
45	118	1
46	74	1
47	62	1

48	75	1
49	74	1
50	60	1

Dari 50 iterasi tersebut dilakukan selama 268.48 detik / 4.4747 menit dengan jumlah tindakan paling banyak 166 tindakan serta tindakan paling sedikit 48 tindakan dengan rata-rata tindakan 72.96 dan dari 50 iterasi tersebut tidak ada yang menabrak obstacle.

Dari hasil yang di dapatkan perbandingan waktu yang dibutuhkan oleh *Monte Carlo* dan *Linear Function Approximation*.

Tabel 4.53 Waktu Monte Carlo

Persegi	Lingkaran	Huruf
5.9 menit	6 menit	6.2 menit

Tabel 4.54 Waktu Linear Function Approximation

Persegi	Lingkaran	Huruf
3.7 menit	6.3 menit	4.4 menit

Terlihat pada tabel waktu yang dibutuhkan oleh *linear function approximation* lebih efisien daripada *Monte Carlo*

4.7 Integrasi Islam

Integrasi Al-Qur’an yang berkaitan dengan bab 1 adalah tentang kehidupan manusia di dunia. Integrasi Al-Qur’an pada kehidupan manusia di dunia adalah konsep penting dalam ajaran agama islam yang mencakup berbagai kehidupan sehari-hari. Islam memberikan panduan dan pedoman untuk menjalani kehidupan yang seimbang, berarti, dan berkah di dunia.

Dalam Al-Qur'an manusia dinyatakan dengan kata al~nas (240 kali), al insan (64 kali), al-insu (16 kali) al-basyar (37 kali) bam adam (7 kali) dan khalifah/khalaif (6 kali). Dari ayat-ayat al-Qur'an yang menggelar tentang manusia dapat direkam beberapa hal yaitu: 1. Kejadian dan tugas manusia 2. Manusia sebagai makhluk berpikir dan merasa 3. Manusia sebagai makhluk beragama[14].

Manusia diciptakan Allah SWT, melalui pentahapan, yaitu dari nuthfah, kemudian menjadi 'alaqah, dari 'alaqah menjadi mudghah (segumpal daging), dari mudghah menjadi tulang-belulang, kemudian tulang itu dibungkus dengan daging, lalu dijadikan makhluk yang lain

Di sisi lain terkait dengan diciptakannya manusia terdapat informasi di dalam Al Qur'an juga mengenai masa masa waktu di dalam rahim hingga diluar rahim, dengan ayat Al Qur'an nya sebagai berikut.

يٰٓاَيُّهَا النَّاسُ اِنْ كُنْتُمْ فِيْ رَيْبٍ مِّنَ الْبَعْثِ فَاِنَّا خَلَقْنٰكُمْ مِّنْ تُرَابٍ ثُمَّ مِنْ نُّطْفَةٍ ثُمَّ مِنْ عَلَقَةٍ ثُمَّ مِنْ مُّضْغَةٍ مُّخَلَّقَةٍ وَّغَيْرِ مُخَلَّقَةٍ لِّنُبَيِّنَ لَكُمْۗ وَنُقِرُّ فِى الْاَرْحَامِ مَا نَشَاۤءُ اِلٰٓى اَجَلٍ مُّسَمًّى ثُمَّ نُخْرِجُكُمْ طِفْلًا ثُمَّ لِتَبْلُغُوْٓا اَشُدَّكُمْۚ وَمِنْكُمْ مَّنْ يُّتَوَفّٰى وَمِنْكُمْ مَّنْ يُّرَدُّ اِلٰٓى اَرْذَلِ الْعُمُرِ لِكَيْلَا يَعْلَمَ مِنْۢ بَعْدِ عِلْمٍ شَيْـًٔاۗ وَتَرَى الْاَرْضَ هَامِدَةً فَاِذَآ اَنْزَلْنَا عَلَيْهَا الْمَاۤءَ اهْتَزَّتْ وَرَبَتْ وَاَنْۢبَتَتْ مِنْ كُلِّ زَوْجٍۢ بَهِيْجٍ

"wahai manusia! Jika kamu meragukan (hari) kebangkitan, maka sesungguhnya Kami telah menjadikan kamu dari tanah, kemudian dari setetes mani, kemudian dari segumpal darah, kemudian dari segumpal daging yang sempurna kejadiannya dan yang tidak sempurna, agar Kami jelaskan kepada kamu; dan Kami tetapkan dalam rahim menurut kehendak Kami sampai waktu yang sudah ditentukan, kemudian

Kami keluarkan kamu sebagai bayi, kemudian (dengan berangsur-angsur) kamu sampai kepada usia dewasa, dan di antara kamu ada yang diwafatkan dan (ada pula) di antara kamu yang dikembalikan sampai usia sangat tua (pikun), sehingga dia tidak mengetahui lagi sesuatu yang telah diketahuinya. Dan kamu lihat bumi ini kering, kemudian apabila telah Kami turunkan air (hujan) di atasnya, hiduplah bumi itu dan menjadi subur dan menumbuhkan berbagai jenis pasangan (tetumbuhan) yang indah" Qs Al Hajj : 5.

Allah menyampaikan kepada seluruh hamba tentang ayat-ayat kauniyah: "Jika kalian meragukan kekuasaan Kami untuk menghidupkan kalian kembali setelah kematian kalian, maka lihatlah awal mula penciptaan kalian agar keraguan itu hilang. Kami telah menciptakan Adam dari tanah, kemudian Kami menciptakan keturunannya dari air mani yang dipancarkan laki-laki ke rahim perempuan. Kemudian dengan kekuasaan Allah, air mani itu berubah menjadi segumpal darah, kemudian berubah menjadi segumpal daging yang mirip dengan daging yang terdapat bekas gigitannya; terkadang penciptaan ini sampai pada tahap yang sempurna, dan terkadang tidak sampai sempurna. Hal ini agar Kami menjelaskan besarnya kekuasaan Kami.

Dan Kami menjadikan janin yang sempurna penciptaannya tetap di dalam rahim hingga datang masa kelahiran; kemudian Kami mengeluarkan janin ini sebagai bayi yang lemah badan dan panca indranya, yang secara berkala meningkat kekuatannya. Dan sebagian kalian meninggal pada masa muda, dan sebagian lain diberi umur panjang hingga usia lanjut sampai orang ini tidak mengetahui sama sekali apa yang dahulu dia ketahui.

Dan hal lain yang menjadi bukti kebenaran hari kebangkitan adalah tanah yang tandus jika Kami turunkan kepadanya air hujan, maka tanah itu akan mulai ada kehidupan yang bergerak dan mulai menumbuhkan berbagai jenis tanaman yang memberi rasa bahagia bagi orang yang melihatnya karena kecantikan dan keindahannya[14].

4.7.1 **Muamalah *Ma'a* Allah**

Dengan adanya kehidupan manusia di dunia ini hendaknya akan selalu taat kepada Allah dan menjauhi segala larangan yang diberikan, sehingga hidup di dunia akan mendapatkan suatu keberkahan yang akan diberikan oleh Allah seperti yang ada di dalam Al Qur'an Surat At-Talaq ayat 3 sebagai berikut.

وَيَرْزُقْهُ مِنْ حَيْثُ لَا يَحْتَسِبُ ۚ وَمَن يَتَوَكَّلْ عَلَى ٱللَّهِ فَهُوَ حَسْبُهُۥٓ ۚ إِنَّ ٱللَّهَ بَٰلِغُ أَمْرِهِۦ ۚ
قَدْ جَعَلَ ٱللَّهُ لِكُلِّ شَىْءٍ قَدْرًا

" Dan memberinya rezeki dari arah yang tiada disangka-sangkanya. Dan barangsiapa yang bertawakal kepada Allah niscaya Allah akan mencukupkan (keperluan)nya. Sesungguhnya Allah melaksanakan urusan yang (dikehendaki)Nya. Sesungguhnya Allah telah mengadakan ketentuan bagi tiap-tiap sesuatu. " (QS At-Talaq : 3)

Dan menganugerahkan kepadanya rezeki dari arah yang tidak dia duga} tidak terbesit dalam benaknya dan dalam dugaannya (Siapa saja yang bertawakal kepada Allah, niscaya Allah akan mencukupkannya) mencukupkannya (Sesungguhnya Allahlah yang menuntaskan urusanNya) menunaikan ketentuanNya sehingga tidak ada yang luput dariNya dan tidak

ada permintaan yang bisa melemahkanNya (Sungguh Allah telah membuat ketentuan bagi setiap sesuatu) waktu selesainya perkara tersebut[15].

4.7.2 **Muamalah *Ma'a an-Nas***

Dengan adanya kehidupan manusia di dunia ini harus saling bermanfaat satu dengan yang lain dan juga saling tolong menolong untuk kebaikan. Manusia juga diharuskan untuk berinteraksi sesama nya dengan etika dan adab yang baik. Dalam memberikan kebahagiaan tersebut terdapat pada ayat Al-Qur'an surat Al Qashash ayat 77 sebagai berikut.

وَابْتَغِ فِيمَا آتَاكَ اللَّهُ الدَّارَ الْآخِرَةَ ۖ وَلَا تَنسَ نَصِيبَكَ مِنَ الدُّنْيَا ۖ وَأَحْسِن كَمَا أَحْسَنَ اللَّهُ إِلَيْكَ ۖ وَلَا تَبْغِ الْفَسَادَ فِي الْأَرْضِ ۖ إِنَّ اللَّهَ لَا يُحِبُّ الْمُفْسِدِينَ

"Dan carilah pada apa yang telah dianugerahkan Allah kepadamu (kebahagiaan) negeri akhirat, dan janganlah kamu melupakan bahagianmu dari (kenikmatan) duniawi dan berbuat baiklah (kepada orang lain) sebagaimana Allah telah berbuat baik, kepadamu, dan janganlah kamu berbuat kerusakan di (muka) bumi. Sesungguhnya Allah tidak menyukai orang-orang yang berbuat kerusakan." (QS al-Qashash/28: 77).

Dan carilah pahala negeri akhirat pada apa yang Allah berikan kepadamu berupa harta benda, dengan mengamalkan ketaatan kepada Allah melalui harta itu di dunia ini. Dan janganlah kamu lupakan bagianmu dari dunia dengan jalan bersenang-senang di dunia ini dengan hal-hal yang halal, tanpa berlebihan. Dan berbuat

baiklah kepada orang-orang dengan memberikan sedekah, sebagaimana Allah telah berbuat baik kepadamu dengan (memberikan) harta yang banyak. Dan janganlah kamu mencari apa yang diharamkan oleh Allah berupa tindakan berbuat kerusakan di muka bumi dan penganiayaan terhadap kaummu. Sesungguhnya Allah tidak menyukai orang-orang yang berbuat kerusakan dan Dia akan membalas mereka atas amal perbuatan buruk mereka."[16].

4.7.3 **Muamalah *Ma'a al-Alam***

Selain untuk beribadah kepada Allah, manusia juga diciptakan sebagai khalifah dimuka bumi. Sebagai khalifah, manusia memiliki tugas untuk memanfaatkan, mengelola dan memelihara alam semesta. Allah juga telah menciptakan alam semesta untuk kepentingan dan kesejahteraan semua makhluknya maka dari itu manusia harus bisa untuk menjaga serta melestarikan keindahan dari alam semesta ini tanpa merusaknya. Seperti yang tertera pada Ayat Al-Qur'an surat Al A'raf ayat 56 berikut.

وَلَا تُفْسِدُوا فِي الْأَرْضِ بَعْدَ إِصْلَاحِهَا وَادْعُوهُ خَوْفًا وَطَمَعًا ۚ إِنَّ رَحْمَتَ اللَّهِ قَرِيبٌ مِنَ الْمُحْسِنِينَ

Artinya " Dan janganlah kamu membuat kerusakan di muka bumi, sesudah (Allah) memperbaikinya dan berdoalah kepada-Nya dengan rasa takut (tidak akan diterima) dan harapan (akan dikabulkan). Sesungguhnya rahmat Allah amat dekat kepada orang-orang yang berbuat baik ". (QS Al-A'raf : 56)

Berdo’alah (wahai kaum mukminin), kepada tuhan kalian, dengan keadaan penuh menghinakan diri kepadaNya, dengan suara rendah dan perlahan. Dan hendaknya do’a dilakukan dengan hati khusyu dan jauh dari riya. Sesungguhnya Allah tidak menyukai orang-orang yang bertindak melampaui batas syariatNYa. Dan tindakan melampaui batas yang paling besar adalah perbuatan syirik kepada Allah, seperti berdo’a kepada selain Allah, dengan meminta kepada orang-orang yang sudah mati, berhala-berhala dan yang semisalya.[17]

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada penelitian ini, telah dikembangkan sistem *reinforcement learning single agent* dengan dua algoritma pelatihan antara lain *Monte Carlo Methods* dan *Linear Function Approximation.* Sistem dikembangkan dalam upaya berupaya menggerakkan *agent* menuju *target* sambil menghindari rintangan yang mungkin ada di sekitarnya. Tujuan utama agent adalah mencapai target dengan cara memaksimalkan total reward yang diperoleh selama proses perjalanan. Pada penelitian ini, *obstacle* memliki jumlah yang bervariasi antara lain 0,1,2, dan 3. Sedangkan lingkungan juga memiliki bentuk yang bervariasi antara lain persegi, lingkaran, dan bentuk huruf. Berdasarkan pada hasil penelitian, Pengujian dengan menggunakan metode *linear function approximation* terlihat lebih efisien untuk waktu yang digunakan daripada *monte carlo*.

5.2 Saran

Pada penelitian yang akan datang, penulis memberikan beberapa saran di antaranya:

1. Mengkaji efisiensi dan efektivitas masing-masing algoritma pelatihan dalam lingkungan yang lebih kompleks.
2. Melakukan eksperimen dengan mengubah arsitektur jaringan atau parameter lainnya untuk mengamati dampaknya pada kinerja agen.

3. Mengembangkan Penelitian ini pada proses pembuatan game ataupun di dunia nyata.

Dengan memperhatikan saran-saran di atas, penelitian tentang sistem *Monte carlo Method* dan *Linear function Approximation* dengan berbagai algoritma pelatihan dapat memberikan pemahaman yang lebih baik tentang bagaimana mengembangkan agen cerdas yang efektif dan adaptif dalam berinteraksi dengan lingkungan.

DAFTAR PUSTAKA

1. Radiyan Rahim RNF. Aplikasi dalam simulasi penjualan dengan menggunakan metode monte carlo. *Reg Dev Ind Heal Sci Technol Art Life*. 2018;II:235-239.

2. Safrina S, Irfan M. Ability To Read Quran and Understanding of Tajwid for Sriwijaya University Students. *Conciencia*. 2020;20(2):77-84. doi:10.19109/conciencia.v20i2.6486

[3] V. François-lavet *et al.*, "An Introduction to Deep Reinforcement Learning. (arXiv:1811.12560v1 [cs.LG]) http://arxiv.org/abs/1811.12560," *Found. trends Mach. Learn.*, vol. II, no. 3–4, pp. 1–140, 2018, doi: 10.1561/2200000071.Vincent.

[4] G. T. Lee and K. Kim, "A Controllable Agent by Subgoals in Path Planning Using Goal-Conditioned Reinforcement Learning," *IEEE Access*, vol. 11, no. March, pp. 33812–33825, 2023, doi: 10.1109/ACCESS.2023.3264264.

[5] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," *31st AAAI Conf. Artif. Intell. AAAI 2017*, pp. 2140–2146, 2017, doi: 10.1609/aaai.v31i1.10827.

[6] Putri WL. Penggunaan Monte Carlo Untuk Optimalisasi Prediksi Pengadaan Barang Di QShop Batam. *JR J RESPONSIVE Tek Inform*. 2018;2(1):101-108. doi:10.36352/jr.v2i1.130

[7] Alfikrizal K, Defit S, Yunus Y. Simulasi Monte Carlo dalam Prediksi Jumlah Penumpang Angkutan Massal Bus Rapid Transit Kota Padang. *J Inform Ekon Bisnis*. 2020;3:78-83. doi:10.37034/infeb.v3i2.72

[8] Firdaus A. *Implementasi Algoritma Q-Learning Untuk Perencanaan Jalur Mobile Robot*. Univeristas Komputer Indonesia; 2022.

[9] Melo FS, Ribeiro MI. Instituto de Sistemas e Robótica Q -learning with linear function approximation 1 Q -learning with linear function approximation. Published online 2007:1-23.

[10] Jin C, Yang Z, Wang Z, Jordan MI. Provably Efficient Reinforcement Learning with Linear Function Approximation. *Math Oper Res*. 2023;48(3):1496-1521. doi:10.1287/moor.2022.1309

[11] R. Axtell, “Why agents?: on the varied motivations for agent computing in the social sciences,” *Cent. Soc. Econ. Dyn. - Brookings Inst.*, no. 17, pp. 1–23, 2000, [Online]. Available: http://www.brookings.edu/~/media/research/files/reports/2000/11/technology axtell/agents.pdf%5Cnhttp://www.brook.edu/~/media/Files/rc/reports/2000/11technology_axtell/agents.pdf

[12] M. Lapan, *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.

[13] L. F. Huang, *Artificial intelligence*, vol. 4. 2010. doi: 10.1109/ICCAE.2010.5451578.

[14] https://tafsirweb.com/5741-surat-al-hajj-ayat-5.html

[15] https://tafsirweb.com/10983-surat-at-talaq-ayat-3.html

[16] https://tafsirweb.com/7127-surat-al-qashash-ayat-77.html

[17] https://tafsirweb.com/2509-surat-al-araf-ayat-55.html