Implementasi Penyelesaian Permainan Teka-teki Silang dengan Pendekatan Constraint Satisfaction Problem

Irfan Alamsyah
irfan1alamsyah@apps.ipb.ac.id

Rahmad Ilham Sani
haltcrayonilham@apps.ipb.ac.id

Mochammad Kevin Ariobimo
mochammadkevin@apps.ipb.ac.id

Naufal Akbar Rahardjo
naufalakbrahardjo@apps.ipb.ac.id

Andyana Lilmuttaqina Mafaza
andyanamafaza@apps.ipb.ac.id

Department of Computer Science, IPB University, Bogor, Indonesia

***Abstrak* - Perkembangan teknologi yang pesat dalam beberapa dekade terakhir telah menciptakan dampak signifikan pada kehidupan manusia, termasuk di bidang kecerdasan buatan (AI). Salah satu permasalahan dalam pembangunan kecerdasan buatan adalah *Constraint Satisfaction Problem* (CSP). CSP memerlukan suatu nilai yang dipilih dari domain terbatas yang diberikan, untuk dimasukkan ke setiap variabel dalam suatu permasalahan hingga seluruh kendala yang terkait dengan variabel dapat dipenuhi. Dengan menerapkan metode penyelesaian CSP pada permainan teka-teki silang, dapat secara langsung melihat bentuk objektif permasalahan CSP dalam bentuk praktik. Proyek ini bertujuan untuk menerapkan CSP pada permasalahan teka-teki silang. Penyelesaian CSP dilakukan dengan memerhatikan dan memanfaatkan konsep *constraint progation* dengan algoritma AC-3 dan konsep *backtracking* dengan heuristik *minimum remaining values*. Penerapan CSP dapat dilakukan pada permasalahan teka-teki silang dengan beberapa pertimbangan terkait dimensi papan dan algoritma yang digunakan.**

Kata kunci - constraint satisfaction problem, teka-teki silang, backtracking, constraint propagation

I. PENDAHULUAN

Perkembangan teknologi yang pesat dalam beberapa dekade terakhir telah menciptakan dampak signifikan pada kehidupan manusia. Salah satu bidang yang diperkirakan dapat berdampak pada kehidupan manusia adalah memahami cara manusia berpikir yang disebut *Artificial Intelligence* (AI). AI diciptakan tidak hanya untuk memahami namun serta menciptakan entitas cerdas [1]. Dalam rentang waktu enam bulan (Juni hingga November 2023), terdapat lebih dari 2000 perangkat lunak di bidang teknologi pemasaran, dan sekitar 73% dari perangkat lunak tersebut berbasis AI [2].

Salah satu implementasi entitas cerdas adalah yang dapat menemukan solusi terhadap suatu permasalahan secara efisien. *Constraint Satisfaction Problem* (CSP) adalah salah satu permasalahan yang dapat diselesaikan ketika seluruh variabel memiliki nilai yang memuaskan seluruh batasan [1]. CSP dapat diungkapkan dalam bentuk berikut. Diberikan himpunan variabel, beserta himpunan nilai yang mungkin untuk diberikan kepada setiap variabel, dan kumpulan batasan, hingga temukan nilai variabel yang memuaskan seluruh batasan [3].

Teka-teki silang dapat memberikan lingkungan pengujian yang menarik untuk evaluasi CSP [4]. Teka-teki silang sering digunakan sebagai contoh pada permasalahan CSP dan dapat digunakan sebagai pembuatan dalam teka-teki silang itu sendiri [5].

Penelitian ini bertujuan untuk mengimplementasikan penyelesaian teka-teki silang dengan pendekatan CSP. Sebelumnya, penelitian terdahulu oleh Connor J. *et al.* (2005) [4] telah mengimplementasikan metode seperti *forward checking, dynamic value ordering, conflict-directed backjumping, arc consistency,* dan berbagi optimisasi lainnya. Sementara itu, penelitian sebelumnya oleh Ernandes M. *et al.* (2005) [6] lebih fokus pada pemecahan masalah teka-teki silang dapat batas waktu 15 menit. Penelitian ini khususnya berfokus pada implementasi pemecahan teka-teki silang dengan menggunakan pendekatan CSP dengan propagasi *constraint* dan *backtracking* dengan algoritma AC-3 dan heuristik MRV.

Implementasi tersebut akan diberikan pemahaman teka-teki silang dan batasan yang perlu dipahami. Pengerjaan ini dilakukan dengan harapan dapat mendapat wawasan terhadap penerapan kecerdasan buatan. Penemuan cara dan kompleksitas terhadap algoritma yang digunakan. Memahami dan meningkatkan kemampuan sistem untuk mengatasi tantangan linguistik dan logika yang kompleks. Keberhasilan dalam mengaplikasikan kecerdasan buatan pada TTS tidak hanya relevan dalam konteks permainan kata-kata, tetapi juga memiliki implikasi luas pada pengembangan sistem cerdas untuk memahami dan memanipulasi bahasa dalam berbagai bentuk aplikasi.

II. TINJAUAN PUSTAKA

A. Kecerdasan Buatan

Kecerdasan Buatan (AI) merujuk pada sub disiplin ilmu komputer yang terfokus pada pengembangan mesin dan program komputer yang memiliki kemampuan menyerupai proses berpikir manusia dan memanfaatkan kemampuan untuk memproses data, melalui pembelajaran dari latihan, serta mengambil keputusan secara mandiri [7]. Dalam konteks AI, terdapat kemampuan untuk melakukan pemrosesan data dan pembelajaran mesin, serta kemampuan untuk memecahkan masalah kompleks dan mengambil keputusan berdasarkan informasi yang diproses. Lingkup AI mencakup pula pengenalan pola dan citra, memungkinkan sistem untuk mengenali dan menafsirkan data dalam berbagai bentuk seperti suara, gambar, atau pola kompleks. Saat ini, AI juga dirancang untuk berinteraksi dengan manusia melalui berbagai antarmuka, baik suara, teks, maupun visual, yang dapat ditemukan dalam aplikasi seperti asisten AI dan chatbot. Terobosan terkini dalam AI terus membawa dampak signifikan di berbagai sektor industri, dan perannya di masa depan diantisipasi akan semakin penting.

B. Constraint Satisfaction Problem

Constraint Satisfaction Problem (CSP) adalah algoritma pencarian yang memanfaatkan struktur *state* atau kondisi dan menggunakan tujuan secara umum dibandingkan spesifik pada permasalahan heuristik untuk memungkinkan penemuan solusi permasalahan kompleks. Tujuan utama dari ide pemikiran ini adalah untuk mengeliminasi sebagian besar porsi dari ruang pencaharian secara bersamaan dengan mengidentifikasi kombinasi variabel atau nilai yang melanggar batasan [1]. Beberapa kategori dalam CSP memiliki algoritma penyelesaian yang nihil untuk efisien, yakin adalah permasalahan yang bersifat NP-complete. Dalam praktiknya, sudah cukup untuk menemukan solusi dengan biaya komputasi yang wajar, yang memenuhi sebagian besar kendala, terutama apabila permasalahan terdapat batasan yang "mudah" dan objektif. Jika semua atau sebanyak mungkin batasan yang terpenuhi, maka dapat disebut sebagai solusi eksak; sebaliknya, maka disebut pendekatan [3]. CSP terdiri dari tiga komponen utama. *X* (variabel) adalah himpunan variabel, *D* (domain) adalah himpunan domain yang mewakili domain nilai yang dapat diambil oleh masing-masing variabel, dan *C* (*constraint*) adalah himpunan batasan yang menentukan kombinasi nilai yang diperbolehkan.

C. Teka-teki silang

Teka-teki silang adalah sebuah permainan yang terdiri dari baris dan kolom kotak yang dapat diisi dengan suatu huruf dengan pola horizontal atau vertikal dengan setiap kotak diberikan warna yang berbeda, yaitu hitam dan putih [8]. Pemain teka-teki silang harus mengisi setiap kotak putih dengan huruf yang sesuai dengan pertanyaan yang diberikan. Teka-teki silang merupakan permainan yang dapat melatih otak dan dimainkan oleh banyak orang [9].

III. METODE

Penelitian ini menggunakan dua langkah utama, yaitu Propagasi *Constraint* dan *Backtracking Search*. Propagasi *Constraint* fokus pada pembatasan nilai variabel, dan mengimplementasikan *local consistency* seperti *node-consistent* dan *arc*-consistent dengan algoritma AC-3. Backtracking Search menggunakan *depth-first search* (DFS) dan algoritma heuristik *Minimum Remaining Values* (MRV) untuk mencapai solusi dengan konsistensi lokal dan efisiensi algoritma yang optimal.

A. Propagasi Constraint

Constraint adalah relasi yang membatasi atau mengkondisikan nilai yang dapat dimiliki oleh suatu variabel. Kendala tersebut dapat melibatkan nilai dari variabel lain [10]. *Constraint* dapat dikategorikan kedalam beberapa kategori. *Unary constraint* merupakan batasan yang mengikat satu variabel. *Binary Constraint* mencakup batasan yang menghubungkan dua variabel. Untuk menyelesaikan CSP, diperlukan definisi *state space* dan konsep solusinya. Setiap *state* dalam CSP didefinisikan sebagai pemberian nilai kepada beberapa atau seluruh variabel. Pemberian nilai yang tidak melanggar batasan disebut *consistent*. Propagasi constraint dilakukan sebelum proses pencarian dengan cara *local consistency*. Proses dari *local consistency* mengakibatkan nilai yang tidak konsisten dapat dihilangkan.

Node-consistent adalah salah satu tipe dari *local consistency* yang akan tercapai ketika semua domain dari variabel mematuhi *unary constraint*. Disisi lain, *Arc-consistent* akan terjadi ketika semua domain dari variabel tersebut memenuhi *binary constraint*. Algoritma AC-3 digunakan untuk mencapat *arc consistency*. Untuk mencapai seluruh variabel telah *arc-consistent*, algoritma AC-3 menjaga *queue arc* untuk dipertimbangkan [1]. AC-3 beroperasi pada *constraint*, variabel, dan domain variabel (ruang lingkup). Sebuah variabel dapat mengambil salah satu dari beberapa nilai diskrit; himpunan nilai untuk variabel tertentu disebut sebagai domainnya. Jika terdapat inkonsistensi, langkah-langkah koreksi diterapkan untuk menjaga konsistensi global. Kombinasi dari *node consistent* dan *arc consistent* digunakan untuk mencapai *local consistency*, dimana suatu kondisi domain yang tidak diperlukan dapat dihilangkan untuk meningkatkan efisiensi algoritma. Dengan demikian, pendekatan ini bertujuan untuk memastikan bahwa setiap variabel mematuhi *unary constraint* ataupun *binary constraint*, sehingga domain yang tidak relevan dapat diminimalisasi dan membuat solusi dengan keadaan lokal yang lebih konsisten.

B. Backtracking Search

Backtracking search digunakan dalam program untuk *depth-first search* (DFS) yang memilih nilai variabel satu per satu dan *backtrack* ketika variabel tidak memiliki nilai untuk diberikan. *Backtracking search* secara terus menerus akan memilih variabel yang belum ditelusuri dan mencoba memasukan semua nilai di domain pada variabel, berusaha mendapatkan solusi. Jika terdapat ketidakkonsistenan ditemukan, maka akan *backtrack* dan menunjukkan "gagal", menyebabkan panggilan sebelumnya untuk mencoba nilai lainnya. *Backtracking search* hanya menyimpan satu representasi dari state dan merubahnya dibandingkan membuat yang baru.

Minimum Remaining Values (MRV) adalah algoritma heuristik yang bertujuan untuk mengurutkan variabel dengan nilai legal yang paling sedikit. MRV umumnya memberikan kinerja yang lebih baik dibandingkan dengan pengurutan acak atau statis, terkadang dengan faktor 1.000 atau lebih, meskipun hasilnya bervariasi secara signifikan tergantung pada masalah yang dihadapi.

C. Metode Pengerjaan

Pengerjaan proyek penelitian ini dilaksanakan oleh 5 anggota dengan masing masing memiliki peran dalam pembangunan proyek. Pengerjaan dilaksanakan dalam ruang waktu selama lima minggu dimulai pada tanggal 10 November 2023. Dengan rangkaian urutan pengerjaan sebagai berikut:

Waktu	Pengerjaan
10 - 12 November 2023	Mengumpulkan informasi serta seluruh yang diperlukan untuk topik yang disediakan untuk menentukan topik yang mampu untuk dikerjakan.
13 - 22 November 2023	Perangkaian program TTS generator dan TTS Solver, testing performa dan ketepatan dari program secara iteratif.
23 - 30 November 2023	Merapihkan seluruh komponen program, bug fixing, dan pembangunan web demonstrasi.
1 - 10 December 2023	Mengumpulkan data dan hasil untuk Penyusunan laporan dan PPT.

Fig. 1. Timeline pengerjaan proyek penelitian.

Pengerjaan proyek dilaksanakan secara *remote* menggunakan *github collaborator*. Program yang dibangun menggunakan bahasa pemrograman Python untuk *solver*, *script*, dan *generator*. Program visualisasi dengan memanfaatkan web yang secara keseluruhan menggunakan bahasa pemrograman HTML yang didukung dengan

Bootstrap. Program yang sudah pada tahap finalisasi akan kemudian di diuji dengan perangkat komputer Lenovo V14G2ITL.

IV. HASIL DAN PEMBAHASAN

A. Implementasi Program

Program solver berfungsi secara otomatis dalam menyelesaikan teka teki silang menggunakan pendekatan *Constraint Satisfaction Problem* (CSP) dan algoritma *backtracking*. Program akan membaca papan teka teki silang dan daftar kata dari *file*, kemudian menciptakan variabel dan mengonfigurasi *constraint* yang ada antar variabel. Selanjutnya, dengan menjalankan algoritma AC-3 serta *backtracking*, *solver* akan mencari solusi yang memenuhi semua *constraint*. Hasil akhirnya adalah papan teka teki silang yang terisi lengkap dengan kata-kata yang sesuai dengan aturan dan *constraint* yang diberikan.

Program akan menginisialisasi *dictionary* 'assignment' yang akan digunakan untuk menyimpan nilai yang telah dimasukkan ke dalam variabel selama proses pencarian solusi. Kemudian, *solver* akan membaca *file* "crossword.txt" yang berisi domain kata dan melakukan konfigurasi untuk konsistensi agar membuat semua kata dalam *list* menjadi huruf kapital.

```
def main():
  assignment = {}
  boardstr = read_file("crossword.txt")
  words = read_file("words.txt").splitlines()
  words = [word.upper() for word in words]
```

Fig. 2. Membaca file kumpulan kata dan menyatarakannya sebagai huruf kapital.

Selanjutnya adalah membuat daftar variabel 'V' yang menggunakan fungsi 'create_variables' dan menerima parameter papan teka teki silang 'boardstr' yang diambil dari program generator dan kata kata yang akan diisi nantinya 'words'. Fungsi ini juga menerapkan *node consistency* dengan menghilangkan domain dengan panjang kata yang tidak sama dengan panjang variabel yang diinginkan. Lalu dibuat sebuah himpunan *arc* 'S' antar variabel menggunakan fungsi create_arc. Arc inilah yang akan digunakan nantinya untuk penerapan algoritma AC-3.

```
V = create_variables(boardstr, words)
S = create_arc(V)
```

Fig. 3. Pembuatan variabel V dan S untuk menyimpan parameter papan serta himpunan arc.

Fungsi 'arc_consistency_3' adalah implementasi dari algoritma AC-3 yang bertujuan untuk mengurangi domain variabel-variabel dalam CSP dengan memastikan konsistensi pada himpunan *arc* yang menghubungkan variabel. Fungsi akan menerima parameter 'S' yang merupakan himpunan *arc* dan merepresentasikan hubungan antar variabel-variabel. Proses AC-3 diimplementasikan untuk memastikan bahwa setiap nilai dalam domain variabel dapat memenuhi aturan constraint yang ada. Fungsi ini akan memanggil fungsi 'revise' yang memiliki parameter 'X', 'Y', dan 'Cxy' di dalamnya, dimana 'X' adalah domain secara horizontal, dan 'Y' vertikal. Fungsi tersebut akan memeriksa dan memperbaharui domain variabel 'X'. Jika terdapat nilai dalam domain 'X' yang tidak memenuhi *constraint* yang berlaku atau tidak 'Cxy', maka nilai tersebut akan dihapus dari domain 'X'. Proses ini akan berulang hingga tidak ada revisi yang dapat dilakukan pada arc tersebut atau hingga domain variabel menjadi kosong, aturan ini juga berlaku pada 'Y' untuk domain secara vertikal.

```
def revise(Vx, Vy, Cxy):
  if not Cxy:
    return False
  revised = False
  elements_to_remove = set()
  for x in Vx.domain:
    satisfied = False
    for y in Vy.domain:
      if x[Cxy[0]] == y[Cxy[1]]:
         satisfied = True
         break
    if not satisfied:
      elements_to_remove.add(x)
      revised = True
  Vx.domain.difference_update(elements_to_remove)
  return revised

def arc_consistency_3(S):
  for s in S:
    X, Y, Cxy = s
    revise(X, Y, Cxy)
```

Fig. 4. Algoritma AC-3 untuk menghapus nilai yang tidak memenuhi pada domain.

Fungsi 'backtrack' menerima parameter 'V' dan *dictionary* 'assignment'. Fungsi ini memiliki *base case* dengan memeriksa apakah panjang variabel sudah sama dengan jumlah variabel 'V', jika iya maka semua variabel telah dimasukkan nilainya dan fungsi mengembalikan 'True' sebagai tanda bahwa solusi telah ditemukan. Variabel dipilih menggunakan fungsi 'select_unassigned_variable' yang menggunakan algoritma heuristik MRV untuk pemilihan variabel yang belum memiliki nilai. Selanjutnya, fungsi 'backtrack' akan melakukan iterasi pada nilai nilai yang ada dalam domain variabel 'Vx'. Jika suatu nilai memenuhi *constraint* dan belum dimasukkan ke variabel lain, nilai tersebut akan ditambahkan ke 'assignment' dan proses rekursif dilanjutkan dengan memanggil fungsi 'backtrack' kembali. Jika nilai dalam domain 'Vx' tidak memenuhi constraint atau jika proses rekursif tidak menghasilkan sebuah solusi, nilai tersebut akan dihapus dari 'assignment' dan proses diulang kembali. Proses akan diulang untuk semua nilai dalam domain 'Vx', dan jika tidak ada nilai yang memenuhi *constraint*, fungsi akan mengembalikan 'False'.

```
def backtrack(V, assignment):
  if len(assignment) == len(V):
    return True
  Vx = select_unassigned_variable(V, assignment)
  for val in Vx.domain:
    if val in assignment.values():
      continue
    if satisfy_constraint(V, assignment, Vx,
val):
      assignment[Vx] = val
      reduce_domain(V, assignment, Vx, val)
      result = backtrack(V, assignment)
      if result:
        return True
    assignment.pop(Vx, None)
    restore_domain(V, assignment, Vx, val)
  return False
```

Fig. 5. Algoritma backtracking berjalan secara rekursif dengan base case assignment sudah terisi penuh.

Fungsi ‘select_unassigned_variable’ menerima parameter ‘V’ dan *dictionary* ‘assignment’. Fungsi ini menjalankan prinsip heuristik MRV dengan memilih variabel dengan jumlah domain yang tersisa yang paling sedikit.

```
def select_unassigned_variable(V, assignment):
  unassigned = []
  for v in V:
    if v not in assignment:
      unassigned.append(v)

  unassigned.sort(key=lambda x: len(x.domain))
    return unassigned[0]
```

Fig. 6. Algoritma MRV untuk menentukan variabel pada backtracking

B. *Pengujian Program*

Dimensi	Tipe (ms)			
	AC-3 dan MRV	***MRV, tanpa AC-3***	***AC-3, tanpa MRV***	***Tanpa AC-3 dan MRV***
4x4	0.0283	0.1163	0.0345	0.1609
5x5	0.0173	0.0614	0.0094	0.088
6x6	0.0235	0.1257	0.011	0.1745
7x7	0.0359	0.1963	0.0425	0.1932
8x8	0.0615	0.4024	0.058	0.4415
9x9	0.0565	1.5128	0.0456	0.3145
10x10	0.0471	0.7828	0.0628	0.9363
11x11	0.0654	1.9157	0.1555	2.6235
12x12	0.0995	4.7125	0.2671	3.2786
13x13	0.1335	0.8523	0.1257	80.3709
14x14	0.1461	793.0685	0.1768	2.7889
Rata-rata	0.065	73.0679	0.0899	8.3064
Simpangan baku	0.0437	238.801	0.0814	23.9316

Fig. 7. Performa program

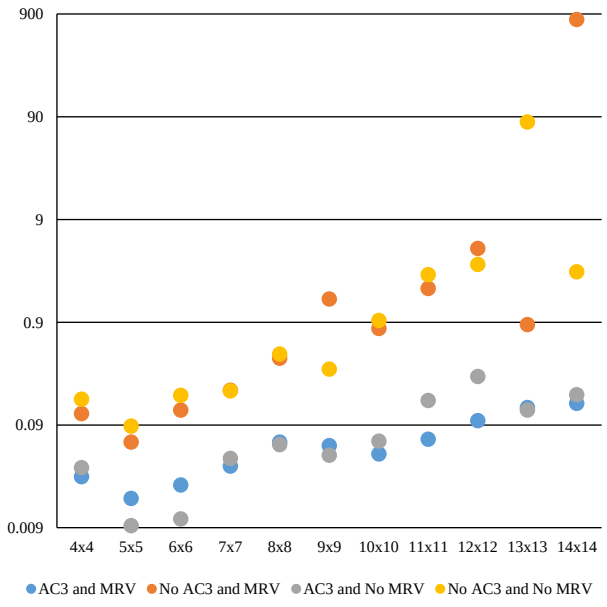

Fig. 8. Ilustrasi Scatter plot performa program. (*Sumbu y disajikan dalam skala logaritmik dengan basis 10)*

Penelitian ini melibatkan pengumpulan data waktu rata-rata dengan 100 kali percobaan untuk setiap sel. Setiap dimensi melibatkan 10 papan yang berbeda dengan ukuran yang sama. Sebagai contoh, untuk dimensi 4x4, digunakan 10 papan berukuran 4x4 yang memiliki bentuk berbeda. Setiap papan diuji 10 kali. Semua papan yang digunakan dapat ditemukan solusinya dengan tingkat okupansi di atas 0.5 yang dikalkulasikan dengan (1). Penggunaan domain mencakup 1200 kata bahasa Inggris yang diambil secara acak, dengan distribusi sebanyak 100 kata untuk setiap rentang panjang kata, mulai dari 3 hingga 14 huruf.

$$Tingkat\ Okupansi = \frac{Total\ kotak\ putih}{Total\ kotak} \quad (1)$$

Analisis data menunjukkan bahwa baris rata-rata menggambarkan kecepatan program, di mana nilai yang lebih tinggi mencerminkan kinerja yang lebih lambat. Selain itu, nilai simpangan baku mengukur konsistensi waktu eksekusi program, di mana nilai yang lebih tinggi mencerminkan variasi yang lebih besar antara percobaan.

C. *Evaluasi Program*

Dari hasil pengujian yang dilakukan, terlihat bahwa penerapan algoritma AC-3 secara tunggal menghasilkan peningkatan yang signifikan dalam kecepatan. Efek serupa terlihat pada penggunaan MRV sendiri, meskipun kinerjanya kurang optimal dibandingkan dengan AC-3 dalam beberapa kasus di mana program mengalami penurunan kecepatan secara mencolok. Namun, ketika kedua algoritma, AC-3 dan MRV, digunakan secara bersamaan, terjadi peningkatan yang signifikan dalam kecepatan dan konsistensi program. Hal ini berdampak positif pada efisiensi pencarian solusi dalam CSP.

Analisis data menunjukkan bahwa kombinasi AC-3 dan MRV secara konsisten memberikan kinerja lebih baik dalam menyelesaikan teka-teki silang. Temuan ini mengungkap strategi efektif untuk menangani CSP dalam konteks aplikasi tersebut. Integrasi AC-3 untuk mengurangi domain variabel dan MRV untuk pemilihan variabel yang optimal terbukti memberikan hasil yang signifikan, membuka peluang peningkatan pengalaman pengguna dan efisiensi dalam penyelesaian permainan. Temuan ini dapat menjadi dasar untuk pengembangan strategi penyelesaian CSP yang lebih optimal dalam berbagai aplikasi.

V. SIMPULAN

Penyelesaian teka-teki silang dapat dilakukan dengan pendekatan permasalahannya sebagai CSP. Kata-kata pada papan didefinisikan sebagai variabel. sementara daftar kata didefenisikan sebagai domain. Panjang suatu kata berfungsi sebagai *unary constraint* dan hubungan antar kata sebagai *binary constraint*. Penyelesaian pada CSP ini dapat diawali dengan *constraint propagation* dan pencarian solusi dapat dilakukan dengan metode *backtracking.*

Pada implementasi pemecahan masalah teka-teki silang dengan pendekatan CSP dengan algoritma AC-3 dan MRV, terlihat keberhasilan yang signifikan dalam menemukan solusi yang optimal. Pendekatan ini menunjukkan keseimbangan yang baik antara kecepatan pencarian solusi dan ketepatan dalam menangani kompleksitas teka-teki silang. Dengan menggunakan AC-3 dan MRV, proses pencarian solusi menjadi lebih cepat dan efisien, bahkan mampu mengatasi tantangan dalam menyelesaikan teka-teki silang dengan ukuran yang besar. Hasilnya, program ini menjadi lebih adaptif dan dapat diandalkan, membuka potensi penggunaan

dalam skenario teka-teki silang yang beragam. Kesuksesan implementasi ini memberikan kontribusi positif terhadap pengembangan solusi untuk permasalahan CSP, khususnya dalam konteks teka-teki silang.

DAFTAR PUSTAKA

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River: Prentice-Hall, 2010.

[2] K. Davis, "AI driving an exponential increase in marketing technology solutions," *Martech*, December 5, 2023. [Online], Available: https://martech.org/ai-driving-an-exponential-increase-in-marketing-technology-solutions/. [Accessed December 8, 2023].

[3] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," *European Journal of Operational Research*, vol. 119, no. 3, pp. 557–581, 1999. doi:10.1016/s0377-2217(98)00364-6

[4] Connor J, Duchi J, Lo B. Crossword Puzzles and Constraint Satisfaction. Stanford University

[5] M. L. Ginsberg, M. Frank, M. P. Halpin, and M. C. Torrance, "Search Lessons Learned from Crossword Puzzles," In Proc. The Eight of National conference on Artificial intelligence, 2004, 210-215.

[6] M. Ernades, G. Angelini, and M. Gori, "WebCrow: a WEB-based system for CROssWord solving," *Association for the Advancement of Artificial Intelligence*, pp. 1412-1417, 2005.

[7] Karman, Strategi dalam Mengembangkan Teknologi Kecerdasan Buatan, Majalah Semi Ilmiah Populer Komunikasi Massa, vol 2, 2021.

[8] Rahayu AC, Maria YWL, Sophia TC. 2021. The Use of Crossword Puzzle in Teaching English Vocabulary. English Teaching, Literature, and linguistics (Eternal).

[9] Cahyo AN. 2011. Gudang Permainan Kreatif Khusus Asah Otak Kiri Anak. Jogjakarta: Flashbooks.

[10] A. K. Mackworth. Consistency in Networks of Relations. Artificial Intelligence, pp: 8:99-118. 1977.