A^*_r: a Bounded Suboptimal Search algorithm using Conditional Node Re-expansion Policy

Tien Minh Dam, Dao Tran Le, Long Viet Truong,
Huyen Thi Dinh, Ngan Kim Nguyen, Tuan Anh Nguyen, Hung Viet Bui, Nguyen Manh Tiem, Le Manh Ha
Viettel High Technology Industries Corporation, Vietnam
{tiendm9, daolt3, longtruong, huyendt40, ngannk6, tuanna63, hungbv, tiemnm, halm8}@viettel.com.vn

***Abstract*—The A* algorithm is a well-known graph search technique widely used in various domains like robotics, games, and logistics. A* combines breadth-first search and heuristics to find optimal paths by evaluating cost and heuristic estimates. However, optimization challenges persist, leading to the need for suboptimal solutions within predefined quality bounds. In the case of Bounded Suboptimal Search, a possible strategy is to reduce unnecessary re-expansions effectively. Existing researches have explored distinct approaches across two primary domains: Never Re-expansion (NR) and Always Re-expansion (AR). In this paper, we introduce an innovative paradigm termed Conditional Re-expansion (CR). This novel algorithm selectively permits re-expansions that yield a considerable decrease in the *g-cost* value, contingent upon an acceptance error parameter denoted as r. Additionally, we propose an efficient anytime algorithm to rectify the outputs of the algorithm mentioned before. This involves prioritizing the repair of states that were denied re-expansion in the CR phase. Theoretical groundwork and practical demonstrations across diverse benchmarks have been conducted to validate the findings.**

Index Terms*—A* algorithm, path planning, bounded suboptimal search, weighted A

I. Introduction

In the realm of problem-solving, particularly in the context of planning, almost real-world scenarios demand the efficient navigation of spaces, routes, or networks. Such instances encompass a wide spectrum of applications, ranging from robotics and autonomous vehicles to logistics and computer communication. The field of computer science has offered invaluable techniques to address these challenges, with heuristic search algorithms playing a pivotal role in devising optimal solutions.

The classic A* algorithm [1] and its variant, Iterative Deepening A* (IDA*) [2], have proven at solving heuristic search problems optimally. As demands for versatility and adaptability have grown, the Weighted A* (WA*) algorithm [3] emerged, allowing us to balance between solution quality and computational efficiency through a weighted cost function. Building upon this foundation, the concept of Revised Dynamically Weighted A* [4] was introduced. This innovative extension to WA* dynamically adjusts the weight assigned to each node's estimated distance, resulting in improved search efficiency while maintaining solution quality. The evolving landscape of heuristic search led to the development of Bounded Suboptimal Search (BSS) [5]. It incorporates a cost parameter within the bound calculation, thus offering a more flexible approach to balancing solution quality and computational resources.

It is a widely acknowledged fact that A* utilizing a consistent heuristic, signified by *f-cost* exhibiting monotonic growth along paths, solely expands a state subsequent to the discovery of the most cost-effective path leading to it. Nonetheless, this principle is not applicable to scenarios featuring non-monotonic f functions, wherein f may decrease along certain paths. Such situations arise when the employed heuristic h is inadmissible or even if it is admissible but inconsistent. In these instances, a state, denoted as s, could be generated with a lower *g-cost* value subsequent to its prior expansion. This circumstance necessitates the removal of s from the $Close$ list, which contains states that will not be considered for expanding, and its subsequent re-inclusion in the $Open$ state list, where it can be expanded again. This procedure is referred to as reopening. A re-expansion, on the other hand, arises when the state, having been extracted from the $Open$ list, undergoes a secondary expansion subsequent to its earlier expansion. Note that states reopened will be re-expanded later if the algorithm runs out of the $Open$ list. The act of re-expansion is discretionary, allowing for a choice of not expanding. This practice often entails updating its *g-cost* value and its parent pointer as appropriate [6]. Two fundamental reopening policies constitute the baseline approach: Always Re-expansion (AR) and Never Re-expansion (NR).

AR and NR policies represent the polar ends of the spectrum, advocating either the constant re-expansion or perpetual avoidance of node re-expansion. However, a more adaptable approach can be realized through the implementation of a hybrid policy. This strategy affords the flexibility to selectively reopen specific nodes, while concurrently opting to abstain from reopening others, thus striking a balance between these contrasting paradigms.

Looking at these situations, this paper introduces two novel algorithms aimed at continuing to enhance the performance of WA* and it can be widely applied in other BSS algorithms.

- The first algorithm A^*_r, implements Conditional Re-expansion (CR) Policy, selectively re-expands state when a shorter path is found. This approach reduces the number of node re-expansions, leading to improved efficiency. The algorithm is specified by a r value.
- The second algorithm, named the Incremental Repair A^*_r , addresses cases where the founded solution lacks

TABLE I
DOMAINS WHERE WA* PERFORMS POORLY DUE TO THE COST OF RE-EXPANSIONS

Algo.	Domain	A* (w=1.5)	WA* (w=2)	WA* (w=5)	WA* (w=10)	WA* (w=20)	BFS
Node re-expansions (ratio)	Maze_512	0.25	0.79	0.93	0.95	0.95	**0.96**
	London_1024	0.40	0.74	0.83	0.87	0.89	**0.91**
	Berlin_1024	0.41	0.96	**0.98**	**0.98**	0.97	0.96

quality due to the conditional re-expansion policy by repair strategy.

These algorithms collectively contribute to a more efficient and effective search for optimal solutions in BSS.

II. BACKGROUND

In the domain of BSS, algorithms face the dual challenge of (1) discovering a feasible solution and (2) establishing that the solution lies within the suboptimal bound. A* and WA* effectively accomplishes both objectives through a straightforward priority function. Additionally, the research [8] includes a bounding function $B(c)$, which provides an upper bound on the solution quality for the given problem. For a problem instance with a solution cost C^*, the algorithm's objective is to return a solution P that costs $C \leq B(C^*)$. In the case of WA*, the bounding function $B(c)$ is defined as w-optimal. The scope of this paper is limited to the study of w-optimal bounding functions, as other bounding functions introduce complexities beyond the paper's focus.

To motivate the study of not performing re-expansion, we demonstrate the results of numerous previous researches. Re-expansion and reopening have a significant impact in BSS [9]. Performing or not performing re-expansion also has an impact on solution quality [8]. Martelli's research [10] showed that if an A* search which re-expands nodes is used on a problem that has $G(V, E)$, $|V| > 4$ as a graph, then the search will require $O(2^{|V|})$ expansions to explore that region of the search space. Martelli proposed a modification to A* which only requires $O(|V|^2)$ expansions to traverse through such graph while still ensuring that all solutions found will still be optimal. However, by adopting NR strategy, recent research of Valenzano et al. [6] ensured that the maximum required expansions are capped at $|V|$, thereby minimizing computational overhead. Yet, it is important to note that this approach comes without a reduction of guaranteed optimality.

This is illustrated in Table I which shows the substantial ratio of re-expansion nodes can lead to undesired search performance of AR. We examine the performance of WA* with different settings by changing the weight of heuristic. Best-First Search (BFS) is the special case of WA* when $w = \infty$. As can be seen in the table, the large effort of expansion is truly for re-expansion, and as a result, it downgrades the performance of WA*.

A simple but effective approach to resolve issues of WA* re-expansions is NR. We experimented both AR and NR in real world domains (table II) and found that NR outperformed AR in almost all cases. However, NR has a significant issue: general BSS NR may fail to return a solution within the bound of quality (exception of WA*). Recent work NRR1, NRR2 [11] attempt to fix the issues of BSS NR by restarting repair from previously failed NR. Note that in the mentioned paper [11], authors used the term NR, AR meaning to Never Reopening, Always Reopening, but theory behind them is the same with re-expansion [6]. NRR2 implements a repair mechanism by using an Inconsistent list (ICL). The general idea of maintaining ICL and adding it to the $Open$ list at a later stage was used before in the context of anytime search algorithms. It was first proposed by [7] as part of their Anytime Repairing A* (ARA*) algorithm, where a series of WA* executions are performed with NR. After every iteration, the w parameter is decreased and the ICL is added to $Open$ list.

III. THEORETICAL NOTATION

This section provides theories, assumptions and notions used consistently in the theoretical parts of this paper.

Termination and completeness. Theoretically, on finite graphs G with non-negative edge weights, A* is guaranteed to terminate and is complete, i.e. it will always find a solution (a path from start to goal) if one exists. On infinite graphs with a finite branching factor and edge costs that are bounded away from zero $d(x, y) > w > 0$, A* is guaranteed to terminate only if there exists a solution. Realistically, construction a full graph G sometimes costly (computation, memory) and a minor part of G takes part in the search. We just only care some specific areas $G'(V', E') \mid V' \subset V, E' \subset E$ in G.

Consistency. If path planning problem is consistent, $\langle A, G(V, E), s_s, s_g \rangle$ unchanged and the A* algorithm have constant heuristic, A* algorithm give the same solution for all run, except incremental A*. We consider the consistency of heuristic function: (1) $h(s_g) = 0$ and (2) $h(s_i) \leq c(s_i, successor(s_i)) + h(successor(s)), \forall s_i \neq s_g$.

Optimality and admissibility. The heuristic function $h(s_i$ is called admissible if $h(s_i$ is never larger than $c^*(s_i)$, namely $c^*(s_i)$ is always less or equal to true cheapest cost from s_i to the s_g. A* is admissible if it uses an admissible heuristic, and $h(s_g) = 0$. If the heuristic function $h(s_i)$ always underestimates the true cost $h(s_i) \leq c^*(s_i)$, A* is guaranteed to find an optimal solution.

Suboptimality. A cost function $c : E \rightarrow R^{>0}$ assigns a cost to each edge. The cost of a path is the sum of the edges along that path. A bounded suboptimal problem is then given by $\langle A, G(V, E), s_s, s_g \rangle$ where $w \geq 1$. (or ϵ in some cases). C^* is the length of an optimal solution path (i.e., one with lowest cost) The cost function will be $f(s) = g(s) + w * h(s)$. Such solutions are said to be w-optimal or w-admissible.

Notation of ***g-cost*** **error**. The value measured the difference between the true *g-cost* value and the *g-cost* value after the

TABLE II
COMPARISON OF NR AR ON STREET MAP BENCHMARK DOMAINS

Criterion		A* AR	A* NR	BFS AR	BFS NR
Solution length	Boston_1024	1546	1629	1632	1633
	Shanghai_1024	1487	1532	1614	1624
	NewYork_1024	1468	1540	1542	1548
Average time (ms)	Boston_1024	7367	3764	26	22
	Shanghai_1024	3145	1316	24	20
	NewYork_1024	3504	1817	58	51

(re-)expansion t was first described in [6] and denoted as *g-cost* error. Let $g^*(s)$ is the true optimal *g-cost* value of the state s. After the re-expansion s, if the $g_t(s)$ value is lower than the previous value of $g(s)$, the re-expansion will be valid. The *g-cost* error was referred to $g_t^\delta = g_t(s) - g^*(s)$.

To convenient remainder, in this paper, the similar term *g-cost* coverage-error mentioned the subtraction of the tentative *g-cost* value after re-expansion t from the existed *g-cost* value after (re-)expansion $t-1$, was denoted as $g_{t-1}(s) - g_t(s)$. The behavior of *g-cost* coverage-error is never enlarged as the search progresses because re-expansion only happens if *g-cost* is better updated.

IV. CONDITIONAL RE-EXPANSION POLICY

The first innovative contribution of this paper is a conditional re-expansion policy (CR) as described by Algorithm 1. In AR, the algorithm re-expands to old states whenever it finds a better solution to this one. In NR, if a node is moved from the $Open$ list to the $Close$ list, it will not expand the second time. The main idea of NR is to increase the performance of the search by ignoring all the re-expansion. CR is the algorithm that lies on the gap between AR and NR. By only doing potential re-expansions, it reduces the number of unnecessary re-expansions. The condition of re-expansion is if the current re-expansion repairs the large enough error to this current state (as determined by the comparison of the*g-cost* coverage-error to the r value - line 16 in Algorithm 1). We denote r as the *g-cost* error acceptance bound that gives a threshold for *g-cost* coverage-error that will be fixed.

To motivate the use of CR, this section describes why the hybrid re-expansion approach can be helpful. Initially, the idea of the CR policy originates from an observation made while examining the substantial volume of re-expansions occurring within a single node. This concept was triggered by insights gleaned from analyzing the distribution of re-expansion occurrences in the wide range of cases. Figure 1 illustrates the frequency of re-expansion efforts observed during a the BFS operation conducted on the Berlin street map dataset [12]. Furthermore, within diverse practical domains, it becomes evident that only a few re-expansions have the potential to substantially enhance solution quality. In the best situation, the algorithm only endorses (re-)expansions featuring the lowest *g-cost* values. Nonetheless, the present search process lacks the capability to determine the ideal (re-)expansion, so the better-enough one is inherently the suitable choice.

Figure 2 illustrates the ECDF chart of the *g-cost* coverage-error when running BFS AR in the same domain as Figure

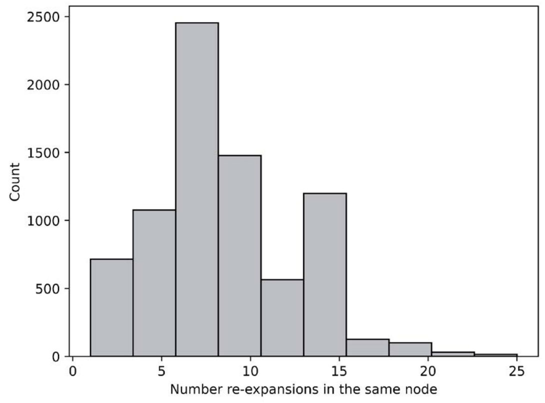

Fig. 1. Frequency of number re-expansions in the same node

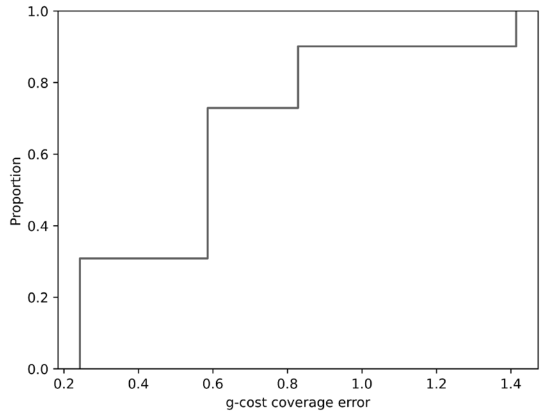

Fig. 2. Proportion or count of observations falling below each unique value in tuple of g-cost coverage error

1. The cumulative value of errors surpassing 0.6 constitutes a mere 20% of the overall error count, yet remarkably contributes to 60% of the total errors. Within more intricate domains, this value demonstrates a substantial increase. Notably, when experimenting with numerous practical cases, we found that fixing a minor number of *g-cost* coverage-errors can substantially improve the quality of the search by reducing a large amount of total *g-cost* coverage-error.

Lemma 1 about the completeness of CR: *In a finite state space, at all times, a bounded suboptimal search (i.e. BFS, WA*) with consistent heuristic is complete even if it refuses all re-expansions.*

Although it is practically demonstrated by the performance

Algorithm 1 Generic A^*_r

```
procedure CR_ASTAR(start, goal, r)
    $Open \leftarrow \{start\}$                                ▷ Priority Queue
    $Close \leftarrow \varnothing$
    while $Open \neq \varnothing$ do
        $s_i, g(s_i) \leftarrow Open.remove\_best()$
        if $s_i = goal$ then return $Solution$
        end if
        $add\ s_i\ to\ Close$
        for each $s_j \leftarrow successor(s_i)$ do
            $g(s_j)_{new} \leftarrow g(s_i) + c(s_i, s_j)$
            if $s_j\ is\ in\ Open$ then
                update new cost $g(s_j)_{new})$
                set $s_i$ as parent $of\ s_j$
            else if $s_j\ is\ in\ Close$ then
                set $s_i$ as parent $of\ s_j$
                if $g(s_j)_{old} - g(s_j)_{new} > r$ then
                    move $s_j$ from $Close\ to\ Open$
                end if
            else
                $Open.add(s_j, g(s_j)_{new})$
            end if
        end for
    end while
end procedure
```

of the best first search without re-expansion, we restate this lemma to show the proof of CR termination.

Proof: In the best case, the CR algorithm needs n expansions (as the length of a solution with the smallest number of states). In the worst case, because of no re-expansion, this algorithm has to expand to all states before finding the goal s_g then terminate.

A* with a consistent heuristic is guaranteed not to expand any state more than once [7]. This is also true in the case of conditional re-expansion. Nevertheless, the search is without the bound for the inconsistent heuristic [6].

V. Incremental Repair A^*_r

The second contribution of this research is the introduction of a more advanced version of the Anytime A* search, denoted as Incremental Repair A^*_r (IRA^*_r) and outlined in Algorithm 2. This abstract pseudo code is motivated by the framework of NRR2 [11] but introduces considerable refinements. Firstly, the Inconsistent list is replaced by a distinct set termed *Repair* (line 4). While the concept of reserving nodes for later use, as seen in Anytime Repairing A* [7], is retained, this novel set focuses on maintaining exclusively the most promising re-expansions, specifically selecting the reduction of the most substantial *g-cost* coverage-errors. Secondly, the prioritization of repairs based on *g-cost* coverage-error values ensures a more efficient implementation, especially in the case of time-bound Anytime A* search. By rectifying larger errors ahead of smaller ones, the algorithm is poised to rapidly converge towards a provable solution. The algorithm concludes under two conditions: upon the attainment of a provable solution or in cases where no promising nodes remain available (the *Open* list is empty)

In practical scenarios, the IRA^*_r algorithm demonstrates remarkable convergence efficiency, typically requiring only few loops to approach convergence. This is achieved by setting a much lower value for r (line 16) in the next loop, effectively reducing the scope for acceptance of errors. This approach yields results closely aligned with, if not surpassing, the (sub-)optimal solutions achievable through the AR policy.

Algorithm 2 Generic Incremental Repair A^*_r

```
procedure REPAIR_COND_REEXP_ASTAR(start, goal)
    $Open \leftarrow \{start\}$                                ▷ Priority Queue
    $Close \leftarrow \varnothing$
    $Repair \leftarrow \varnothing$
    $r \leftarrow inital\ value$
    while $Open \neq \varnothing$ do
        $Sol$, $Repair \leftarrow CR\_Astar(start, goal, r)$
        if $Sol\ is\ provable$ then return success
        else
            $Open \leftarrow Open \cup Repair$
            $Close \leftarrow Close \setminus Repair$
            $r \leftarrow lower(r)$
        end if
    end while return failure
end procedure
```

As a consequence of a finite state environment, this algorithm always terminates when the solution was found or all the state is (re-)expansion. In the worst case, the heuristic is very inconsistent or the environment contains several local minima leading the algorithm travels through $2k$ nodes (k is the number of vertices of the graph) after it finds the goal.

Lemma 2 about the completeness of IRA^*_r: *In a finite state space, at all times, if there is at least a provable solution, IRA^*_r always finds one.*

Proof: As the consequence of the existence of a provable solution, at the base case $r = 0$, the CR algorithm turns to AR, so it always meets the requirement for this lemma. Let r_{min} is the minimum *g-cost* error of all nodes. When the line 12 was called many times, hence, r decreases to r_{min}, it also meets the condition.

In another way, if the distance of loosest provable solution and the latest found solution in particular space is major, the amount of error acceptance bound should be set bigger, so r is down considerably.

VI. Experimental results

To demonstrate the results of the new innovative algorithms, we performed and compared them on several benchmarking domains.

The classical domains: Firstly, we evaluated our approach on classical domains including the blocks world (BW), and sliding puzzle (STP). These famous domains, while well-studied, serve as a foundational baseline to showcase the

TABLE III
THE TREND OF RE-EXPANSION AND OPTIMALITY OF WA* AND BFS WITH DIFFERENT RE-EXPANSION POLICIES

Map Set	Avg. Num. States	Heuristic Acc.	Branching factor	Avg. Re-expansion Ratio						Avg. Optimality					
				A* ($w = 1.5$)			BFS ($w = \infty$)			A* ($w = 1.5$)			BFS ($w = \infty$)		
				AR	CR	NR	AR	CR	NR	AR	CR	NR	AR	CR	NR
BW	59,049	0.29	2	0.12	0.01		0.46	0.3		0.88	0.86	0.82	0.26	0.25	0.23
STP 3x3	362,880	0.33	[1-3]	0.02	0.00		0.50	0.34		0.88	0.88	0.81	0.50	0.49	0.46
Road network	5,172,069	0.44	[0-12]	0.19	0.00		0.58	0.15		0.90	0.89	0.82	0.71	0.70	0.67
Random-40	96,365	0.46		0.06	0.00	0.00	0.78	0.60	0.00	0.90	0.89	0.87	0.84	0.83	0.73
Mazes-32	253,819	0.49	[0-7]	0.27	0.00		0.97	0.87		0.99	0.97	0.92	0.99	0.93	0.79
Street-1024	1,048,576	0.54		0.43	0.00		0.83	0.55		0.99	0.97	0.96	0.88	0.87	0.85
BG512	73,930	0.76		0.31	0.00		0.66	0.42		0.97	0.96	0.92	0.91	0.89	0.85

effectiveness of our algorithms. The heuristic is designed as simplest as possible, while still being admissible and effective. All of the domains are finite state problems with some different characteristics:

The grid-based domains: The test set is taken from Moving AI Lab [12], which is widely used to benchmark graph searching algorithms. This domain contains maps of various sizes, from 10 x 10 to 1024 x 1024 maps. The branching factor of the problem is 8, and the depth of optimal solution can be 1500 for a 1024 x 1024 map. The heuristic function is implemented by using Euclidean distance.

The road network domain: To introduce a more intricate problem domain inspired by real-world routing challenges, we extend our experimentation to include a novel benchmark domain that we have created. In addition to this, we explore the Graph-based transport network derived from OpenStreetMap. This network encompasses a substantial scale, boasting approximately 5.1 million vertices and 6 million edges. These vertices predominantly represent terminal nodes of pathways, while the edges correspond to segments connecting pairs of nodes. The branching factor within this domain is around 4, signifying the average number of successors a node may have. This expanded framework enables us to delve deeper into the complexities of routing problems and provides a fertile ground for evaluating the efficacy and robustness of our proposed algorithms. The heuristic function measures the orthodromic distance of two points on the sphere which can be calculated by Haversine formula.

Summary This observation in Table III showcases the completely effectiveness of CR in maintaining a balanced trade-off between solution quality and search efficiency. In our dedicated testing framework, we observed a consistent pattern in the behavior of AR, CR, and NR. We employed two criteria for evaluation: (1) **re-expansion ratio** quantifies the overall proportion of re-expansions in relation to the total number of node expansions; (2) **optimality** assesses the quality of the found solution in comparison to the optimal solution. The initial experiment centered around the WA* algorithm $w = 1.5$ while varying its re-expansion policies. Notably, we observed that adopting a higher value of r in A^*_r likely led to superior performance in comparison to both WA* with AR or NR. This advantageous outcome was achieved with only a marginal loss in optimality, offset by a significant reduction in re-expansions. The same experiment was provided to investigate the behavior of BFS CR across various domains. In intricate environments, BFS AR consistently demonstrated a notable re-expansion ratio. By employing either the CR or NR policies, BFS exhibited accelerated search completion. Remarkably, while CR introduced a reduction in optimality, this compromise was substantially smaller compared to NR.

To facilitate a more comprehensive investigation, we generated detailed charts for some specific domains. The depiction in Figure 3 illustrated the performance of WA* (with $w = 2$) across different policies encompassing AR, CR at varying r values, and NR. Evidently, a slight compromise in optimality yields a notable enhancement in the overall performance of WA*. We also conducted time measurements for various configurations of WA* but did not show in this paper, due to a strong correlation between the total time taken and the number of expansions.

Furthermore, to showcase the effectiveness of IRA^*_r, we conducted a comparison with different Anytime A* algorithms using the NRR1 and NRR2 repair strategies. Our results highlight that IRA^*_r not only provides provable solutions but also achieves optimality. Figure 4 displays the correlation between the number of node expansions and solution length in the Berlin street map domain, this trend consistent across other domains. NRR1 is quite slow, because it actives AR algorithm from scratch after the NR failed to find solution with the bound. While IRA^*_r may exhibit slower optimal solution discovery after applying the CR policy, it rapidly finds solutions within predefined bound.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we introduced two novel algorithms, A^*_r (A* with Conditional Re-expansion) and IRA^*_r (Incremental Repair A^*_r), aimed at improving the performance of A* and WA* algorithms in the context of bounded suboptimal search. A^*_r implements a Conditional Re-expansion policy, selectively allowing re-expansions based on a specified *g-cost error acceptance bound* (r), effectively reducing unnecessary re-expansions. IRA^*_r extends the concept by introducing a repair mechanism that targets nodes with substantial *g-cost* errors for re-expansion, aiming to convert suboptimal solutions into optimal or near-optimal ones. Experimental results across various benchmark domains, including classical domains, grid-based domains, and road network domains, demonstrated the effectiveness of these algorithms in enhancing solution quality

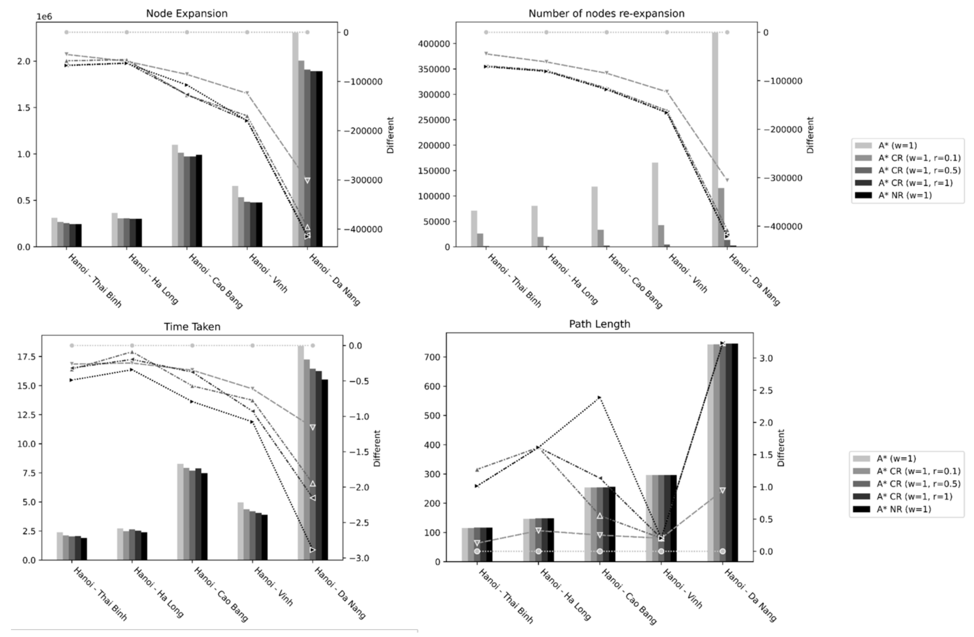

Fig. 3. Performance of WA* AR, WA* NR, A_r^* CR with $w = 2$ on the Road network domain. The bar chart illustrates the net value, while the line chart shows the different value.

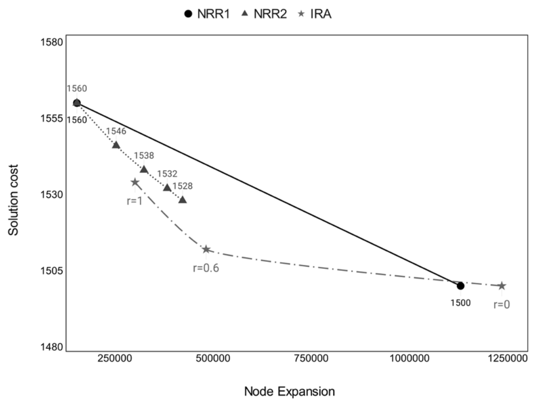

Fig. 4. An illustration of NRR1, NRR2, IRA_r^* performance on Berlin map. IRA_r^* (loose dash line) provides optimal solution after reducing r value from 1 to 0.

and computational efficiency. While the proposed algorithms exhibit promising performance, future work could involve further optimization and fine-tuning of algorithm parameters to adapt to different problem domains and explore their applicability in more complex scenarios. Additionally, research could focus on integrating these algorithms into real-world applications and evaluating their performance in dynamic and uncertain environments.

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Syst. Sci. Cybern., vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[2] R. E. Korf, "Iterative-deepening-A: an optimal admissible tree search," in Proceedings of the 9th international joint conference on Artificial intelligence - Volume 2, in IJCAI'85. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Aug. 1985, pp. 1034–1036.

[3] I. Pohl, "Heuristic search viewed as path finding in a graph," Artif. Intell., vol. 1, no. 3, pp. 193–204, Jan. 1970, doi: 10.1016/0004-3702(70)90007-X.

[4] J. Thayer and W. Ruml, "Using Distance Estimates in Heuristic Search.," presented at the ICAPS 2009 - Proceedings of the 19th International Conference on Automated Planning and Scheduling, Jan. 2009.

[5] J. T. Thayer and W. Ruml, "Bounded suboptimal search: a direct approach using inadmissible estimates," Int. Jt. Conf. Artif. Intell., pp. 674–679, Jul. 2011, doi: 10.5591/978-1-57735-516-8/ijcai11-119.

[6] R. Valenzano, N. R. Sturtevant, and J. Schaeffer, "Worst-Case Solution Quality Analysis When Not Re-Expanding Nodes in Best-First Search," 2016, doi: 10.1609/aaai.v28i1.8850.

[7] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," Jan. 2003.

[8] R. Valenzano, S. J. Arfaee, J. Thayer, R. Stern, and N. Sturtevant, "Using Alternative Suboptimality Bounds in Heuristic Search," Proc. Int. Conf. Autom. Plan. Sched., vol. 23, pp. 233–241, Jun. 2013, doi: 10.1609/icaps.v23i1.13563.

[9] C. M. Wilt and W. Ruml, "When Does Weighted A* Fail?," Symp. Comb. Search, Jan. 2012, doi: 10.1609/socs.v3i1.18250.

[10] A. Martelli, "On the complexity of admissible search algorithms," Artif. Intell., vol. 8, no. 1, pp. 1–13, Feb. 1977, doi: 10.1016/0004-3702(77)90002-9.

[11] V. Sepetnitsky, A. Felner, and R. Stern, "Repair Policies for Not Reopening Nodes in Different Search Settings.," Symp. Comb. Search, pp. 81–88, Jan. 2016, doi: 10.1609/socs.v7i1.18395.

[12] N. R. Sturtevant, "Benchmarks for Grid-Based Pathfinding," IEEE Trans. Comput. Intell. AI Games, vol. 4, no. 2, pp. 144–148, Jun. 2012, doi: 10.1109/TCIAIG.2012.2197681.