An Adaptation Technique to Enhance HTN Planning

Ahmad Salah, Mohamed Nabil Shams El-Din, Ashraf Elsisi
Computer Science Department, Faculty of Computers and Information, Menoufia University, Shebin Elkom, Egypt
ahmad.salah.ahmad.19@gmail.com, m_nabil_shams@yahoo.com, ashrafelsisim@yahoo.com

***Abstract*— There are many and varied areas of the real-world domains that require optimal solutions to their problems. This domains in areas as healthcare, production, public transportation, and driver assistance. Its solutions affect the people involved as well as the organizational efficiency and cost of operations. Recently, research in automated planning optimization has gained importance due to the need for complete automation in such various applications. Hierarchical Task Network (HTN) Planning is hard task even in pure search version and looking for optimal solutions can only be hard. In fact, there are few optimal planners and there are a relatively small number of other planners which usually produce good solutions without guaranteeing the optimality. Therefore, the main objective of this paper is to improve the search technique of HTN planning by adapting Ant system algorithm into refinement planning process by means of plan selection strategy. The proposed search strategy AS-HTN fuses the Ant system to improve to overcome optimal plan challenges. The evaluation of our proposed framework showed an outstanding performance against many plan selection strategies.**

***Keywords*— Hierarchical planning, Ant System Optimization, Automated Planning, PANDA planner, plan selection strategy**

I. Introduction

There are many and varied areas of the real world domains that require optimal solutions to their problems. This domains in areas as healthcare, production, public transportation, and driver assistance. Its solutions affect the people involved as well as the organizational efficiency and cost of operations. Recently, research in automated planning optimization has gained importance due to the need for complete automation in such various applications. However, planning [13] is a very hard task even in pure search version and looking for optimal solutions can only be hard. In fact, there are few optimal planners and there are a relatively small number of other planners which usually produce good solutions without guaranteeing the optimality. Moreover, Hierarchical planning deployed in numerous application domains, but somehow a cause of its formalism it requires planner defined with well structure a cause of search space and solution of the HTN domains. Therefore, it was necessary to apply optimal planning to ensure access to optimal solutions, and accordingly, the Ant System was adapted within the process of producing plans in the Planning and Acting in a Network Decomposition Architecture (*PANDA)* planner [1].

Several new approaches to Hierarchical Task Network (HTN) [2] planning have been introduced in order to optimize the overall performance of HTN planners; however, there are many approaches to solve HTN planning problems, we will mention some related methods from the author's point of view.

Search in plan space search algorithms maintains a partial ordering of tasks during search, these results in less search nodes compared to algorithms that need to commit to a total ordering of tasks. In the classical algorithm, the search starts with an empty plan that is iteratively modified until a solution is found.

Progression search start from initial task and progress in forward direction and applies only to tasks that have no predecessors in the ordering relations. That way, plans are constructed in a forward manner and the planning system has full information about the state. Is less compact representation of the search space since a single partial ordered set may result in a large number of totally ordered sequences.

Translation to propositional logic the search is based on a formalism that is quite different from recent HTN planning. It comes without an initial task/task network, the planner inserts primitive and abstract tasks. The objective is to fulfill a state-based goal; those problems are similar to classical planning problems. Its translation restricted to recursion-free problems and is not able to represent task networks that contain recurring tasks.

Translation to classical planning it simulates a progression search in the state space. A task network is represented in a dedicated part of the state description, and actions are modified to be only executable when they are included in the current task network, and newly introduced actions change that part of the state description according to the decomposition methods. To enable the representation of task networks in the state, the size of intermediate networks generated during search is bounded.

This paper presents the *AS-HTN* plan selection strategy, which is based on fusing Ant Colony Optimization ACO algorithm into PANDA refinement planning process. This approach can be classified as an *adaptive approach*. Among the few works also common on this category was [3], into which messy genetic algorithm adapted beside HTN planning; another work [4] the adaption of metaheuristic Ant Colony Optimization (ACO) with classical planning. However, into our *AS-HTN* such solution built probabilistically step by step, by representing the plan space as a directed graph with partial plans as vertices and plan modifications as edges; ants' movements performing a forward search, each ant choose among different plans (vertices) by means of three functions. First the pheromone value τ (a) function, which returns measurable desirability value of action applicability in state. Second function is η (a) which returns computed value by the heuristic function. The third is Pheromone updating function by which increasing pheromone values for actions used to build the current solution plan and decrease for plans (vertices) that are not involved.

Our strategy was implemented on top of the PANDA planner. Also, an empirical evaluation performed on hierarchical domains, which shows that the proposed strategy outperforms to many other strategies in terms of cases solved.

This paper is organized as follows. Section 2, shortly provide a basis for defining hierarchical HTN planning, and define how planners comprise search control; section 3, discuss the proposed search technique; section 4, show and describe our search algorithm; section 5, provide our experimental results and show comparison results. Finally, section 6 gives notes on possible extensions for the proposed strategy to enhance HTN paradigm.

II. BACKGROUND

The background that is going to be used as the base for our proposed approach is presented in: Automated Planning and specifically into Hierarchical planning and Ant System metaheuristic optimization algorithm.

HTN planning is AI Planning paradigm aims to search into a plan space for a set of predefined tasks, and these tasks constitute the required plan. HTN planning typically includes tasks t, task networks tn, methods m, planning problem P and solution S. Tasks can be either abstract tasks (also named complex tasks) or primitive tasks, tasks are connected through a network (hence the name task network). And methods define how a complex task is decomposed into another task network or primitive task(s). Noting that one task network may be related with more than one method. In order to solve the given planning problem, for HTN planner, the problem specification and domain description are defined as input, and starts performing a the search from the initial task network; however, through the applicability of methods, and the search space grows stepwise until primitive tasks that satisfy the solution criteria are found.

The adaptation approach in this paper is a domain-independent planning framework upon the basis of one by Bernd Schattenberg [5]. The literature has defined a plan space as a directed graph in which vertices are task networks T and edges are refinements m. An outgoing edge m from a vertex T in the plan space is a method for the decomposition of T into another task network. T'= app (m, T). During search, or refinement process, arises a flaw(s) which are defects in the task network that leads to failure in the solution plan. These flaws can arise in the form of (I) abstract tasks, (ii) inconsistent constraints, and (iii) not supported preconditions. As a consequence using a detection function that can compute all flaws of each form in the current task network; then, a triggering function can match those convenient modifications that can be executed in order to solve this type of flaws and finally a modification generating function generates flawless plans according to chosen flaw to be solved by modification selection strategy. To maintain continuity of planning, the planner can choose one plan from different plans resulted from the previous module with the help of a plan selection strategy.

Ant Colony Optimization Algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. This algorithm was proposed by Marco Dorigo in 1992. [5].

Ants will start searching for food from their colony moving randomly in all directions. If an ant finds food returns back towards colony and leave a trail of chemical substances called pheromone along the path. By doing so other ants can detect the pheromone left by the previous ant and they follow the same path. The path is determined by the amount of concentration of pheromone along the path. It is by nature that the pheromone will start evaporating over time reducing the concentration of pheromone thus leading to poor strength of the path. Considering this condition, a shorter path will be suited for ant's. Every ant will follow the shorter path thus keep adding pheromone which makes the concentration strong enough to against evaporation. This makes the emergence of shortest path from colony to food. Seeking a path between the colony and a food source is the natural behavior of ants. Based on this biological behavior, the algorithm was developed aiming to search for an optimal path in a graph.

III. RELATED WORK

Our review describes the algorithmic methods used to obtain optimal solution, or an enhancements in the plan generation process aiming too for optimum solution.

HTN Planning as Heuristic Progression Search [6] propose the use of progression search as basis for heuristic HTN planning systems. Such systems can calculate their

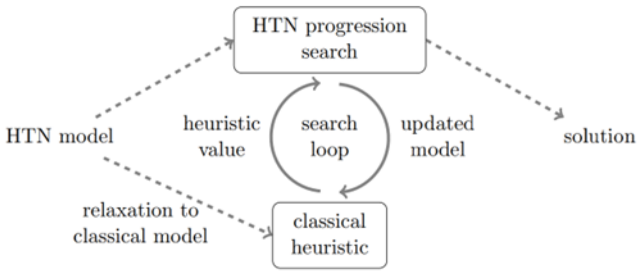

Figure 1 progression search as basis for heuristic HTN

heuristics incorporating the current state, because it is tracked during search. The contribution is the following: Introducing two novel progression algorithms that avoid unnecessary branching when the problem at hand is partially ordered and show that both are sound and complete. Then, introduce a method to apply arbitrary classical planning heuristics to guide the search in HTN planning. It relaxes the HTN planning model to a classical model that is only used for calculating heuristics. It is updated during search and used to create heuristic values that are used to guide the HTN search figure 1. They show that it can be used to create HTN heuristics with interesting theoretical properties like safety, goal-awareness, and admissibility. It relaxes the HTN planning model to a classical model that is only used for calculating heuristics. It is updated during search and used to create heuristic values that are used to guide the HTN search. The empirical evaluation shows that this approach has positive effect on the performance of the overall system.

PYHIPOP-Hierarchical Partial-Order Planner [7] the implementation extends the original POP algorithm [8,9] with an improved preprocessing phase, and adapting the heuristic search for the HTN paradigm. To perform a search in the space of plans, they use different heuristic functions to drive the search. In the first place, a partial plan selection heuristic is used: in figure 2 at line 3.

Algorithm 1: Solve algorithm

```
1 $OPEN \longleftarrow \{top\}$;
2 while $OPEN$ not empty do
3     $n \longleftarrow OPEN.pop()$;
4     if $n.flaws \neq \emptyset$ then
5         return $n.plan$;     // solution found
6     $f \longleftarrow n.flaws.pop()$;
7     for $r \in resolvers(f, n.plan)$ do
8         $OPEN \longleftarrow r$;
9 return $Failure$;
```

Algorithm 1 PYHIPOP–Solving Algorithm

Secondly, a flaw selection heuristic is used at line 6. Flaws are ordered following their kind. They first solve threats, then open links, and eventually expand abstract flaws. In order to sort open links, the implementation uses earliest: the open link from the action coming earlier in the plan is resolved first. Abstract flaws are also sorted using earliest: the tasks coming earlier in the plan are decomposed first.

An admissible HTN planning heuristic [10] the author presents a heuristic based upon the so-called Task Decomposition Graph (TDG) – a representation of the AND/OR structure underlying the task hierarchy. It exploits the TDG to find a least-cost set of primitive actions into which the given abstract tasks can be decomposed. He estimates the effort of refining an abstract task vertex by minimizing over the estimated effort of its method vertices. Analogously, the effort of refining a method vertex can be estimated by summing over the estimates of the tasks it contains. The TDG-recomputation reduces the search space in almost every problem instance, but also increases the runtime for several instances.

Improving hierarchical task network planning performance by the use of domain-independent heuristic search [11] an extended HTN formalism named HTN-e is built so that heuristics from classical planning can be easily adapted to work in the HTN setting. A modified algorithm called OTD-h which combines domain independent heuristics is with ordered task decomposition is proposed. Based on OTD-h and SHOP, a new HTN planning named SHOP-h planning algorithm is established and implemented in Python which is called Pyhop-h. It is a planning algorithm that combines domain independent state-based heuristics with HTN planning. For each task to be decomposed, the planner heuristically selects the best decomposition among the possible ones induced by the set of applicable HTN methods. This technique is domain-independent and can help SHOP and SHOP2 planners to improve their performances. It makes SHOP and SHOP2 planners less sensitive to the order in which the HTN methods appear in the planner's input, making it easier to write HTN domain descriptions.

A generic method to guide HTN progression search with classical heuristics [12] introduce a generic method to apply classical heuristics in HTN planning. They propose the use of progression-based search, because progression algorithms have a concrete state at hand. First, They introduced an improved progression algorithm that increases the performance by reducing the part of the search space searched multiple times. Second, they showed how information about the hierarchy can be incorporated into the non-hierarchical part of the problem. This enables the use of arbitrary state-based classical heuristics to estimate goal distance. The evaluation works on standard (i.e., unchanged) HTN models. It further solves a major problem when using classical heuristics in HTN planning, also this method outperforms state-of-the-art HTN planning systems.

Messy Genetic Algorithm for the Optimum Solution Search of the HTN Planning [3] the author introduces a genetic algorithm to solve the optimum solution searching problem of HTN planning. *Length-variant* chromosome is introduced to represents the possible planning solution in form of decomposition tree with dynamic node numbers. The Genetic algorithm starts with a randomly selected chromosome returned by the Abstract-HTN algorithm as the initial population that encodes a set of possible solution. It is difficult to locate the optimum solution if there is little knowledge for heuristic algorithms to guide the optimum solution search process.

An ACO approach to planning [4] in this approach a stochastically meta–heuristic backward search in the state space of classical planning is proposed, using an Ant Colony Optimization approach. the author is interested on to optimize some quantitative characteristics of a planning problem, as length of plan solutions found. The ants build a plan starting from the initial state s_0 executing actions step by step. Each ant draws the next action **a** to execute in the current state from the set of executable actions. The choice is made by taking into account the pheromone value τ(a) and the heuristic value η(a) for each candidate action with a classical probability transition formula.. They explained that their planner experiments are very encouraging because the system often find better solutions than state of art planning systems in terms of execution time and plan length.

However, improvements to HTN planners have gone in many directions. Each of them has its promising results, and although adaptation approach has admissible results, no one has tried adapting the ant system algorithm with HTN planning.

IV. Proposed Strategy

The proposed framework AS-HTN based as plan selection strategy into refinement plan generation process of PANDA planner, and Ant System Optimization technique. This section presents our proposed plan selection strategy [14] in order to achieve near optimum plan solution using Hierarchical paradigm.

Algorithm 1 illustrate the proposed strategy procedure AS-HTN. Referring to the Ant system approach, take into account heuristic information η about the problem state being solved, which refers to visibility of the problem and means the number of flaws in the current task network. By returning low number of flaws, assigning greater value to η and vice versa. τ i is pheromone quantity deposited into $edge_i$ through which the ants represent their desire to choose from one of the various refinements in the current plan, initial pheromone value is τ_0 = 1.

The Plan transition rule is a probabilistic approach focus on pheromone quantity, deposited on edge, which is the refinements attractiveness and get raised by how many the same refinement was chosen before; and heuristic information "flaw selection strategy" that can guide the ants towards the most promising solutions . After the application of the chosen refinement, measuring the quality of all refinements to be used in pheromone update calculation at the end of each iteration.

```
1 Input: initial plan
2 Output: best
3 Max_tries = 3, max_time =1800 second, $\alpha = 1$, $\beta = 1$, $\rho = 0.5$, $\tau_0 = 1$,
4 $\delta = 1$, $\sigma = 1$, Number of ants n = 4, optimal = 50, best = 0
5 $for_0$ (i=0 ; i < Max_tries; i++){
6   Start_timers (); step = 0; localPlan_Quality = 0; iterP_Quality = 0;
7   Foreach$_1$ ant in n
8         Ant.tour.clear();
9   $End_1$
10  Foreach$_2$ ant in n{
11        Rnd = random(seed) * plan_level_size();
12        Ant.tour[step] = Rnd;
13  }$End_2$
14  While(!((elapsed_time >= max_time) || (best >= optimal ) ) ){
15  Foreach$_3$ ant  n{
16        M= getApplicable_Refinement()
17        Foreach$_4$ edge$_i$ in M {
18             N_flaws$_i$ = ($f_\pi^{det}$ (P, $\pi$))
19             If (N_flaws$_i$ <= 1)
                     $\eta$ = 1/ N_flaws$_i$ + 0.2
20             Else  $\eta$ = 1/ N_flaws$_i$
21             Nominator$_i$ = $\tau_i^\alpha * \eta_i^\beta$
22             Denominatior += Nominator $i$
23  }End4
24  Foreach$_5$ edge $i$ in M
25         Edge_prob $_i$ = Nominator $_i$ ⁄ Denominator
26  $End_5$
27  newP = chooseMaxThenApplyToFringe( Edge_prob $_i$ , $p_i$)
28  ant.tour[++step] = newP
29  iterP_Quality = Q(newP) // Q(newP) = $(\frac{1}{LCF})^{\delta} + (\frac{1}{step})^{\sigma}$
30  If( iterP_Quality > localPlan_Quality )
31         localPlan_Quality = iterP_Quality
32  }End$_3$
33  If(localPlan_Quality > best)
34         best = localPlan_Quality
35  $\tau_{new} = (1 - \rho) * \tau_i + \rho$ *Q(p)
36  } //end while
37  }// $end_0$
38 Return best
```

Algorithm 2 AS-HTN proposed Algorithm

The calculation of plan quality takes account of the heuristic value (e.g. LCF strategy[15]) of refinements, for example, considering that edge1 with $h_{lcf} = 3$ is preferred than edge2 with $h_{lcf} = 5$, since the value returned is the number of flaws to be solved ; the second parameter in calculation is the step into which this refinement was chosen, it's obvious that refinements that satisfy the solution plan in early steps have high quality than others. algorithm [8,9] with an improved preprocessing phase, and adapting the heuristic search for the HTN paradigm. To perform a search in the space of plans, they use different heuristic functions to drive the search. In the first place, a partial plan selection heuristic is used: in figure 2 at line 3.

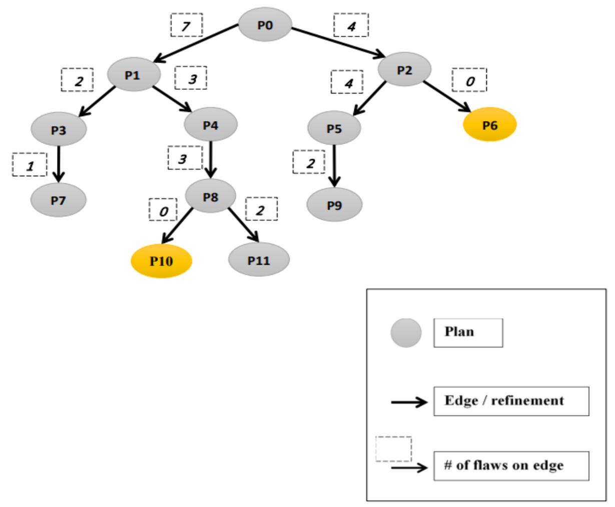

Figure 2 HTN-AS algorithm plan space illustration

If the refinement satisfies the solution raising its quality by one to give it more probability to be chosen in subsequent iterations, δ and σ are parameters for calibrating the preference calculation of each expression. At the end of this step, updating locally the quality value *localPlan* .
Update the quality globally by comparing the local quality, obtained previously, against the best quality obtained from previous iterations. As last step in the algorithm pheromone updating rule into which ρ is the pheromone evaporation rate, and the pheromone deposit for each edge.

Figure 2 represents the plan space, assuming that node $\equiv$ plan and edge $\equiv$ refinement; aiming to choose edges (refinements) that leads to a node solution (food) by using Ant System AS-HTN algorithm.

The algorithm shown in Algorithm 2 starts by initializing the basic variables (lines 3-4), and in subsequent steps it initializes other variables; And before starting search space exploration, clear quality of the local Plan explored in iterations so far (localPlan_Quality) and the plan quality being processed (iterP_Quality) line 6. It then clears Ants' memories at the start of each iteration and allocate for each ant a specified node randomly (line 7 - 13).

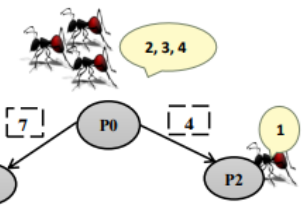

Figure 3 Ant1 choose edge with higher probability

Ants starts building up plans from initial node, P_0 ants [1, 2, 3, and 4]: are in P0; each ant chooses applicable refinements gradually. So, by assuming the execution of the while loop (line 14), follows exploring the available refinements M (line 15), for each ant, and its related flaws determined by modification function f_π^{det} (P, π) (line 18)
Since a heuristic function determines how much promising is the selection of a specific refinement than others; Ant1 has to choose among two refinements (or edges), each has different number of flaws, (P_0, P_1) N_flaws $_1$ =7 and (P_0, P_2)

N_flaws $_2$ =4, figure 3; the heuristic value will be $\eta_1 = {}^1/_7$ $\eta_2 = {}^1/_4$ respectively (line 19 - 20)
Referring to transition probabilities calculation for each edge, using $\tau_0 = 1$, $\alpha = 1$ and $\beta = 1$ Edge_prob$_1 = \frac{1/7}{11/28} = 0.36$, Edge_prob$_2 = \frac{1/4}{11/28} = 0.64$, (line 21 - 26)
Ant1 choose Edge$_2$, which has higher probability
, then adding this edge in its tour path as a new plan step to be: ant1.tour = [P0, P2] (line 27 - 28)
Calculating iteration plan quality of plan in Ant1 iterP_Quality _Quality, assuming LCF strategy returns 6, step = 1, $\delta = 1$ and $\sigma = 1$; since p2 not satisfy solution plan, using equation in (line 29) Q(newP) = $(\frac{1}{1+6})^1 * (\frac{1}{1})^1 = 0.143$, and iterP_Quality = Q (newP) = 0.143
Quality of the Local plan is evaluated at end of every iteration if whether *is greater than* plan quality found so far and localPlan_Quality will be 0.143. (Line 29 - 31)

Repeating the same process steps for Ants [1,2,3]. Ant2 will choose, by using heuristic value $\eta_1 = {}^1/_7$, Edge$_1$ then add it in tour path ant2.tour = [P0, P1]. Also, by measuring plan quality, for example LCF strategy returns 7, step = 1, $\delta = 1$, $\sigma = 1$ and node p1 not satisfy goal (solution plan) Q(newP) = $(\frac{1}{1+7})^1 * (\frac{1}{1})^1 = 0.125$, consequentially iterP_Quality will be compared with localPlan_Quality to select the higher value 0.143.
Ant3 and Ant4 will be have the same calculation as ant [1] and ant [2] respectively, shown in figure 4. Tour paths will be ant3.tour = [P0, P2] and ant4.tour = [P0, P1] Furthermore, localPlan_Quality value will be unchanged and best plan steps founded so far is best = 0.143 (Line 33 - 34).

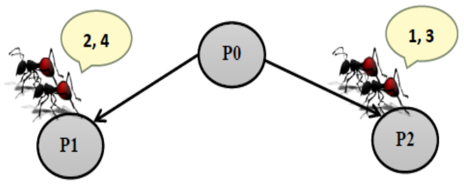

Figure 4 Ants distribution in next level

After ending up ants' transitions and local quality calculations, the pheromone trails update process is done at the end of while loop.

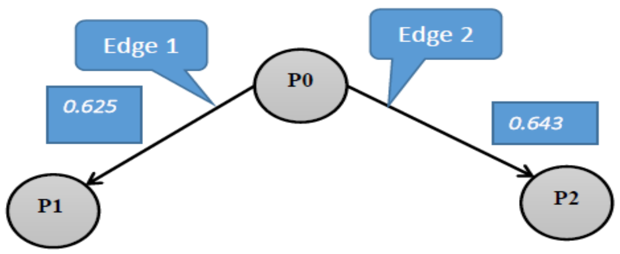

Figure 5 update pheromone quantity

This process starts by evaporating a quantity of pheromone for each traversed refinement (edge) by: $(1 - \rho) * \tau old$ to be 0.5. Next, each ant starts to deposit a determined quantity of pheromone as per tour quality and corresponds to refinements traversed (Line 35). New pheromone quantity on Edge1 τ_{new} = 0.5 + 0.125 = .0625; And for Edge2 τ_{new} = 0.5 + 0.143 = 0.643 as shown in figure 5.

After exploring the plan space level-by-level, ant1 gained the solution plan shown in figure 6, this solution has quality value Δ_1 ($p0$, $p1$, p6) =1.768, which is the optimum plan.

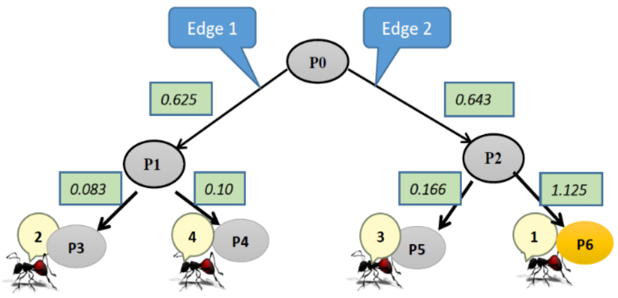

Figure 6 Illustration of ants tour in plan space, and optimum plan gained from ant 1

V. Experimental Results

Our experiments have been performed on HP ProDesk 600 G1 TWR with processor Intel Core i7 4790 GHz and memory of 8 GB. The implementation proved on Java programming language and Eclipse IDE has been used to develop our proposed framework. The implemented approach given is based on the HTN *plan space search* PANDA$_{pss}$. PANDA is a framework that integrate multiple techniques related to HTN planning. PANDA has been developed at the Institute of Artificial Intelligence at Ulm University. The tested version of PANDA called PANDA$_{pss}$, the source code is available on GitHub under an open-source license.
The evaluation efficiency of our strategy over some domains taken from the International Planning Competitions (IPC). In general, these domains are used as standard benchmarks to compare planner performances. A systematic set of computational tests executed over the Satellite and UM-Translog domains. There are other benchmark domains but, they have been chosen among the set of benchmark domains because they offer a variety of problems, also, these problems and its solution plans allow us interesting comments to evaluate our proposed strategy through a course of experiments.
At the first run the AS-HTN plan selection strategy have being tested against 4 different modification selection strategies: (i)HotZone, (ii)Least Committing First (LCF), (iii)Direct Adaptive HotSpot, (iv)Prefer Early Flaws, and the results show that only two plan modification strategies can be promising in the next turn. In fig 7 show that only 2 modification selection strategies that outperforms and have less time than the other. Noting that, the figures show only *UM Translog* domain time since that the time in *Satellite* domain is very short, also elapsed time is converged.
In second turn the computation done again with the same domains and problems but with 6 Plan selection strategies, and 2 modification selection strategies (chosen previously). And the experimental results shows that our strategy AS-HTN out performs the other strategies with *HotZone* strategies shown in figure 8; by means of elapsed time, it requires 4218.306 sec, to return the solution plans for all

this problem. More specifically AS-HTN has solved 11 problems of 16 with the minimum time required.
The second plan selection, in figure 9, was *Smaller Detection Plan Step Ratio*, which requires 4635 sec. This means that AS-HTN is faster by 91%.
By using the second modification selection *Prefer Early Flaws modification*, the AS-HTN strategy out performs again the other strategies; by means of elapsed time, it requires 3097.338 sec, to return the solution plans for all these problems in UM Translog. More specifically AS-HTN has solved 11 problems of 16 with the minimum time required.
The second plan selection was *Smaller Detection Plan Step Ratio*, which requires 3221 sec. This means that AS-HTN is faster by 96%.

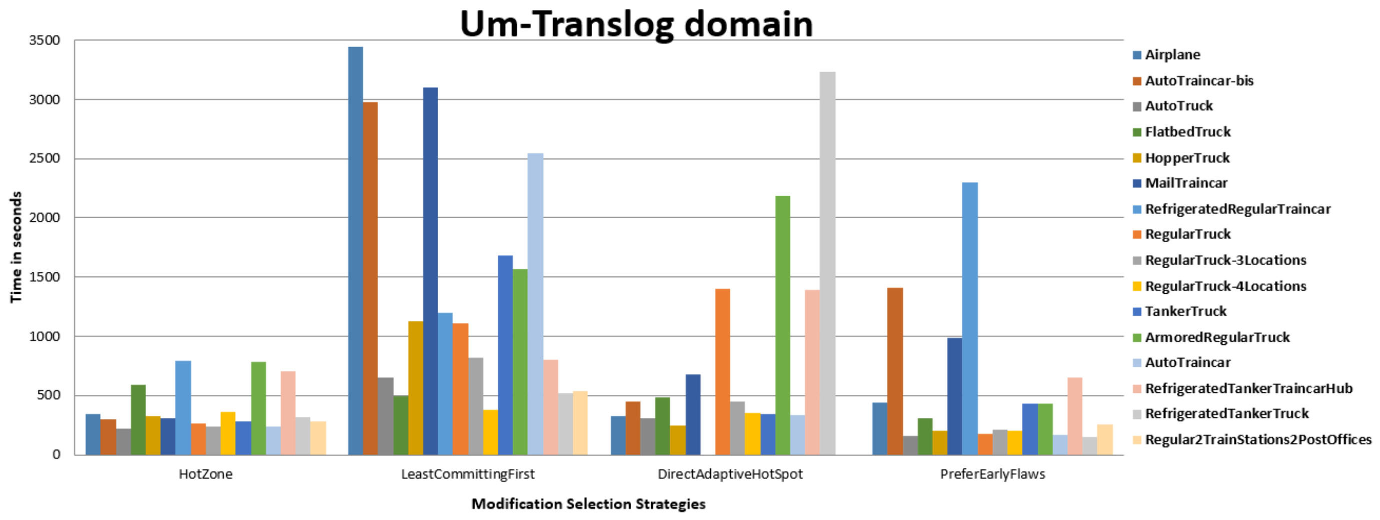

Figure 7 UM Translog domain AS-HTN and 4 modification selections

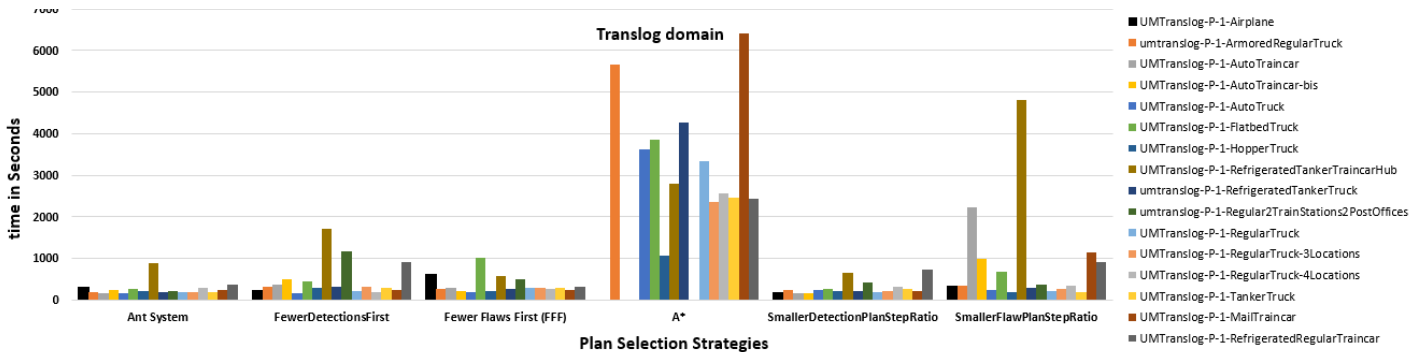

Figure 8 Elapsed time for Um Translog 6 plan selection and Hotzone modification selection

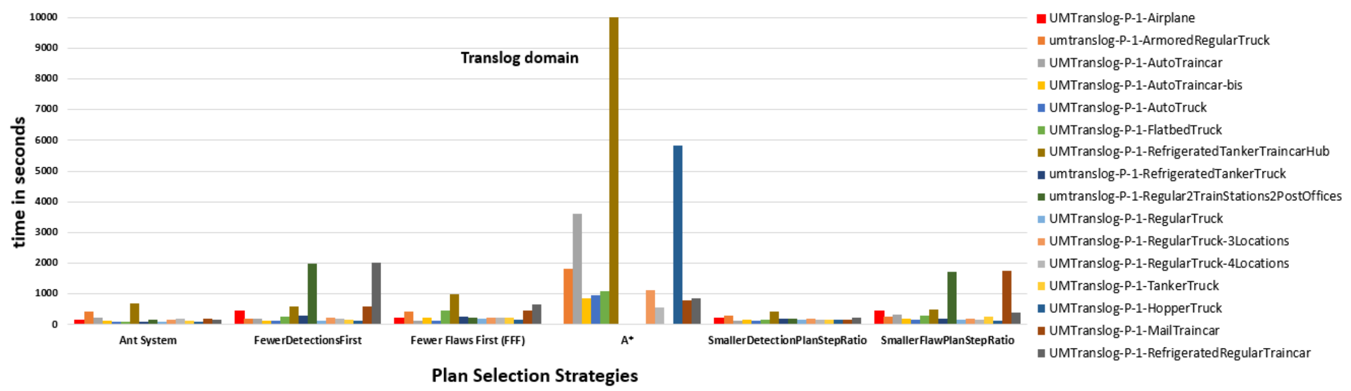

Figure 9 Elapsed time for Um Translog 6 plan selection and prefer early flaws modification selection

VI. CONCLUSION AND FUTURE WORK

Ant System algorithm are powerful when adapted with AI planning, especially with efficient planning approaches like Hierarchical planning. This paper has described a first application of the Ant Colony Optimization meta–heuristic to Hierarchical Planning. The preliminary empirical computation has shown encouraging results and that this approach is applicable for optimization in HTN planning.

For these reasons, this work should be improved and extended in several directions.

First of all, modifying the implementation of the Ant system, and replace it with another Optimization algorithm, also the use of tuning parameters requiring a smaller computation time. Hence it is possible that a less informative and less expensive heuristic function can be used to gain much sensitive performance. Then, another modification is to swap the adaptation of the strategy in the plan generation process: using "*modification selection*" strategy.

References

[1] Daniel Höller, Gregor Behnke, Pascal Bercher, and Susanne Biundo. "The PANDA Framework for Hierarchical Planning." In: Künstliche Intelligenz (KI) 35.3 (2021), pp. 391–396.

[2] Bercher, Pascal, Ron Alford, and Daniel Höller. "A Survey on Hierarchical Planning-One Abstract Idea, Many Concrete Realizations." IJCAI. 2019.

[3] Luo, Jiangfeng, Cheng Zhu, and Weiming Zhang. "Messy Genetic Algorithm for the Optimum Solution Search of the HTN Planning." Foundations of Intelligent Systems: Proceedings of the Sixth International Conference on Intelligent Systems and Knowledge Engineering, Shanghai, China, Dec 2011 (ISKE2011). Springer Berlin Heidelberg, 2012.

[4] Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." IEEE transactions on systems, man, and cybernetics, part b (cybernetics) 26.1 (1996): 29-41.

[5] Schattenberg B (2009) Hybrid planning & scheduling. Ph.D. thesis, Ulm University, Germany

[6] Höller, Daniel, et al. "HTN planning as heuristic progression search." Journal of Artificial Intelligence Research 67 (2020): 835-880.

[7] Lesire, Charles, and Alexandre Albore. "PYHIPOP-Hierarchical Partial-Order Planner." Workshop on the International Planning Competition. 2021.

[8] Penberthy, J. Scott, and Daniel S. Weld. "UCPOP: A Sound, Complete, Partial Order Planner for ADL." *Kr* 92 (1992): 103-114.

[9] Bechon, Patrick, et al. "Using hybrid planning for plan reparation." *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015.

[10] Pascal Bercher, Gregor Behnke, Daniel Holler, and Susanne Biundo. An admissible HTN ¨ planning heuristic. In IJCAI, pages 480–488, 2017.

[11] Cheng, Kai, et al. "Improving hierarchical task network planning performance by the use of domain-independent heuristic search." Knowledge-Based Systems 142 (2018): 117-126.

[12] Höller, Daniel, et al. "A generic method to guide HTN progression search with classical heuristics." Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 28. 2018.

[13] Dana Nau, Malik Ghallab, and Paolo Traverso. Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[14] Schattenberg B, Bidot J, Biundo S (2007) On the construction and evaluation of flexible plan-refinement strategies. In: Proceedings of the 30th annual German conference on AI (KI), pp 367–381. Springer]

[15] Joslin, David, and Martha E. Pollack. "Least-cost flaw repair: A plan refinement strategy for partial-order planning." *AAAI*. 1994.