Adding Flexibility Using Structured Goals: the Case of Itinerant Mobile Agents

Toan Phung,* Seng W. Loke,+ and James Harland*
*School of Computer Science and Information Technology
RMIT University, GPO Box 2476V, Australia
+School of Computer Science and Software Engineering
Monash University, Caulfield East, VIC 3145, Australia
{tphung@cs.rmit.edu.au,swloke@csse.monash.edu.au,jah@cs.rmit.edu.au}

Abstract

We show how a goal-oriented view of mobile agent itineraries separates $what$ itinerary the agent must accomplish (which we call its ITAG-Goals*) from how the agent is to accomplish its goals (i.e., its strategies), thereby providing flexible and robust behaviour in a dynamic network. Our approach generates strategies given a goal and a model of the network environment. We also show how strategies can be reused and how the agent regenerates strategies for its remaining goals in the face of difficulties.*

1 Introduction

We separate an itinerary into $what$ the agent must do and how the agent accomplishes such outcomes. By capturing what must be done without explicitly representing how it is to be done, we allow the mobile agent to become more fault tolerant.

This paper shows how to incorporate the notion of goals into mobile agent itineraries to improve fault tolerance in the face of a changing network.

2 ITAG: The ITinerary AGent Scripting Language

The ITineray AGent (ITAG) language is a scripting language for defining a mobile agent's itinerary, that is, what it must do and where it must do them. ITAG currently contains four operators: Parallel Composition "||", Sequential Composition "$\cdot$", Independent Nondeterminism "|" and Conditional Nondeterminism ":" [7]. Sequential Composition involves a task-by-task execution with each task in the itinerary being dependent on the previous task in the sequence. The Parallel Composition operator allows the agent to perform multiple tasks in parallel. For example, two different mobile agents may perform their own tasks at the same time. This may also be applied to a single agent, where the agent 'splits' and performs more than one task at a time. Independent Nondeterminism behaves much like the logical 'OR' operator: do this task or another task, it doesn't matter which is done. The Conditional Nondeterminism operator is a more stringent version of the Independent Nondeterminism operator whereby if a task can be completed, then the goal has been achieved. But if it cannot be completed, the next task in the itinerary is performed.

An example to illustrate the syntax of ITAG is as follows:

$$A_p^a \cdot A_q^a \cdot A_r^a \cdot A_s^a$$

This itinerary translates into: Move Agent A to Places p, q, r and s and perform Action a at each location [7]. The "$\cdot$" operator represents that the atomic actions are to be performed sequentially from left to right.

The current implementation of ITAG [7] is rigid. If one location where a task is to be performed fails for some reason, the entire itinerary is rendered inoperable. This inflexibility does not allow the agent to deal with errors, failures, security issues or unexpected alterations in the network. ITAG currently has no support for dynamically alterable itineraries - it is not possible to alter the agent's itinerary after the agent has been launched.

3 Mobile Agents with Goal-Oriented Itineraries

By introducing the concept of goals into the mobile agent itinerary, we are able to separate what itinerary the agent needs to achieve (we call such itineraries *ITAG-Goals*) from how it accomplishes this itinerary. In doing this, the agent

Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT'03)
0-7695-1931-8/03 $ 17.00 © 2003 IEEE

can exploit positive changes in its environment, such as cheaper alternative paths and to re-plan around negative changes such as host drop-outs or network traffic congestion. An example of a positive change is when there is a less costly way of getting to a host, either in terms of monetary cost or time. In this event, the mobile agent should be able to re-plan its path using the updated network information.

However, this raises the issue of over-deliberating and wasting resources in a dynamic environment. To deal with this issue, we have introduced the notion of a limit on the amount of pre-processing the mobile agent does. Hence, the agent can plan for a sub-set of its itinerary rather than planning the entire itinerary. The benefit of this is that the agent does not wait until all of its itinerary path has been planned before it begins carrying out its tasks. This allows the agent to reduce the chance of failure and therefore re-planning, particularly in highly dynamic network environments by permitting a more cautious approach.

Goals allow the user to be less specific in what the agent does in order to achieve a particular outcome. The ITAG-Goal language has all the operators in ITAG mentioned above and also another operator, namely, the AND ($\wedge$) operator. This operator allows the agent to complete its goals in no explicit order and hence if a particular network node is the only node that hosts a desired service but is unavailable, the agent should be able to pursue other goals in the meantime instead of wasting time waiting for that node to become available again. The agent can convert the $AND(\wedge)$ operator into either the $Sequential$ or the $Parallel$ operators.

Plans are commonly associated with goals. From abstract goals come plans which are executed to bring about the goal. In our approach, we use the term $strategy$ to refer to sets of low level procedural tasks that are executed by mobile agents to achieve mobility and to achieve ITAG-Goals. If a strategy fails, the agent will devise a new one to replace it in an attempt to fulfill the ITAG-Goal.

With large and complex itineraries, it is important that the agent does not restart from the beginning every time a failure occurs as this leads to the wasting of resources, (e.g., time and money for CPU time). This scenario can be extended to illustrate the concept of *strategy reuse*. If the ITAG-Goal of the agent had consisted of two or more places to travel to and perform operations on, that is, the agent had two or more goals to achieve, the chances of failure between any two goal would also increase. The failure may appear early in the agent's execution cycle or later. The later the failure occurs, the more resources would be wasted if a new agent were to be created to replace the failed agent. Hence, being able to recover from error is of vital importance, resource-wise. Our approach facilitates the $salvaging$ of computed strategies through being able to recompute only the faulty parts of strategies, re-using the rest and continuing from the point of failure.

In ITAG-Goal, the agent views the itinerary as a state of the world that must be brought about in a non-specific way. As an example of how ITAG-Goals are used to promote agent flexibility, consider the goal of travelling to place Z to perform some operation o. Let the starting place of the mobile agent be place X and let the intermediate nodes Y and W be two nodes, each between X and Z. So in effect, there are two paths by which the agent may choose to travel from point X to point Z, which can be represented as: X$\rightarrow$Y$\rightarrow$Z and X$\rightarrow$W$\rightarrow$Z. The agent may have in-built heuristics which allow the agent to select the most appropriate path according to some cost model. For this example, say the agent chooses path X$\rightarrow$Y$\rightarrow$Z. If while travelling along this path there is a failure such as a network link being disconnected, the agent will not be able to complete its goal using its current chosen path. Hence, the agent must re-plan in order to create a workable strategy that still allows it to complete its goal. This strategy is generated dynamically by the agent which involves reflecting back up to the $goal$ level (the itinerary itself) and re-formulating a new travel path. If the problem still exists as a result of there being no other appropriate paths to the goal, then a $goalfailure$ [6] has occurred at which stage, the agent's remaining tasks would be impossible to complete.

ITAG-Goals. ITAG-Goals are given by the user to the ITAG-Goal agent to achieve and remain fixed throughout the agent's lifetime. The exact strategy by which the agent accomplishes these goals is not explicitly defined and as a result, ITAG-Goals provide a level of flexibility through abstraction, a mechanism that goals support. ITAG-Goals can be viewed as a special type of goal in that they are mainly concerned with agent mobility.

There are two types of ITAG-Goals: *atomic goals* and *complex goals*. $Atomic\ goals$ represent the individual itinerary units that are combined to form an overall goal, an atomic goal is defined as A_p^a. It is these atomic goals that are used to generate the strategies. An itinerary may consist of one or more of these atomic goals. The overall goal itself, is the $complex\ goal$. Complex goals are formed by combining atomic goals, e.g., $A_p^a \cdot A_q^b$. The user gives the complex ITAG-Goal to the agent.

In EBNF format, the syntax of ITAG-Goals is the same as in the original ITAG language, namely:

$G ::= 0 \mid A_p^a \mid (G \parallel_{\oplus} G) \mid (G \cdot G) \mid (G \mid G) \mid (G :_{\Pi} G) \mid (G \wedge G)$

"G" represents a goal. "$\cdot$" has the lowest precedence than the other operators (which have equal precedence). The operators are left associative.

Strategies. Strategies represent how the agent is to achieve a particular goal. In executing a strategy, the mobile agent is able to move from one host to another. In general, a

strategy is a set of hosts starting with the agent's current location, a list of intermediate hosts that the agent is to travel to, the target host itself and an associated *cost* with executing that strategy. If the agent reaches a target host and executes the desired operation, then its corresponding atomic goal is complete.

Parallel to ITAG-Goals, we have *atomic strategies* and *complex strategies*. Atomic strategies are specific procedures that are executed in order to accomplish a given atomic goal. Complex strategies are two or more atomic strategies that have been combined to achieve two or more atomic goals. An atomic goal can have more than one strategy. This is because an agent can fail to completely execute a strategy. At this point, a *strategy failure* has occurred and therefore the agent must replan a new migration path (i.e a new strategy) using its new current location and its intended target host.

Strategies encapsulate the travel path and cost. However, this is not all that can be represented. Their representation may also encompass features such as security protocols, network link and host $gain$ [1] and the migration method which may be a $push$, $pull$ [4]. Our approach uses the simplest representation, that is, the strategy's label (for referencing purposes), the set of nodes to travel to (including both the starting and target nodes), and the cost of executing that strategy which means the cost of travelling to all intermediate nodes and finally to the target node.

Atomic strategies will be represented as ${}_hS_{p,q,r}\ (10)$ where S is the strategy's label. The h in front of S represents that the starting point for the journey is from node h (the agent's home). The *p,q,r* represents that the strategy moves the agent to hosts *p*,*q* and r to perform a task at r then returns home to h after its actions have been completed. The '(10)' at the end of the strategy is the cost of executing the strategy both in terms of traversing links and running on foreign hosts. If costs cannot be fully calculated, then approximations can be made. A shortcoming of this representation is that if a complex strategy is formed out of two or more of these atomic strategies, then how will the agent know whether it is currently residing on a goal host or simply a node that will lead it to a goal host? To overcome this, we further define hosts or network nodes as either $target$ or $by-pass$ nodes. Target nodes are hosts that the agent performs tasks on in order to accomplish ITAG-Goals. By-pass nodes are those which are visited only because they lead to target nodes.

In general, the format of the strategies will be ${}_xS_y(C)$ where x is the starting point of the agent and y is a set of one or more hosts that the agent can traverse. The S is a label for the atomic strategy (as there may be more than one strategy for accomplishing an atomic goal) with C representing the cost of executing the strategy.

In converting goals into strategies, some operator constructs need to be preserved. For example, when combining two or more atomic strategies into one complex strategy, how would the agent know when to execute operations sequentially, conditionally or in parallel? This is very clear with ITAG-Goals, as the operators themselves specify the techniques needed and this must also be represented at the strategy level. As a result, we have introduced strategy 'delimiters' that define where sequential, conditional and parallel agent execution is required. In a sense, they tell the agent which execution 'mode' it should be in when executing a strategy. Namely, we have the $SeqStart$, $CondStart$, $ICondStart$ and $ParStart$ delimiters. SeqStart represents that the following set of hosts should be executed sequentially. CondStart represents that the following set of hosts should be executed using conditional non-determinism. ICondStart represents that the following set of hosts should be executed using independent non-determinism. ParStart represents that the following set of hosts should be executed using parallelism. These operators are placed between host names in strategies in order to tell the agent that the next set of host traversals should be done in a particular way. For example, if the agent was given the complex ITAG-Goal: $(A_p^t \parallel A_q^u) :_\Pi A_r^v$, then the corresponding complex strategy might be: ${}_hS_{ParStart}a, b, c, d, P, {}_{ParStart}e, f, g, Q, {}_{CondStart}h, i, j, R(40)$. Note that nodes in lowercase signify that they are by-pass nodes where no important tasks are executed. Nodes in uppercase are target nodes where important tasks (as specififed in the ITAG-Goal) are performed. The way in which this strategy is read and executed is explained later.

The EBNF(Extended Backus-Naur Form) format for strategies is:

$$
\begin{array}{rcl}
Str & ::= & s \mid NULL \\
s & ::= & xSp(cost) \\
p & ::= & ({}_{op}y) \mid p,\ p \\
op & ::= & SeqStart \mid ParStart \mid ICondStart \mid \\
 & & CondStart \mid ANDStart
\end{array}
$$

where x is the name of the node the agent starts on and y is a set of comma separated host names.

Executing Strategies. When an agent has generated all necessary strategies from goals, it must then begin executing them to achieve mobility. To do this we have developed the *ITAG-Goal Interpreter*. The ITAG-Goal Interpreter is an algorithm that encapsulates the agent's life-cycle and behaviour from receiving the complex ITAG-Goal from the user to generating and executing its strategies until completion or failure.

Salvaging Strategies. Complex strategies need not be completely replaced, only the paths which are at fault. The

remaining travel path after the point of failure may still be valid. Hence re-computing the entire complex strategy may be unnecessarily wasteful. Our approach is to replace only the failed path and to $salvage$ the rest of a complex strategy.

In order to salvage portions of the complex strategy, only the current goal that the agent is trying to achieve is needed. Once this has been established, the path between the agent's current location and its intended target host is computed. Any remaining portions of the strategy are reused to create a new complex strategy with the failed path replaced with the new computed path.

As an example, consider the complex strategy ${}_hS_{SeqStart}a, b, c, d, E_{ANDStart}f, g, h, I_{ANDStart}j, k, l, M(40)$. If the agent was executing this complex strategy but found that when it reached node c discovered that a direct path between node c to d no longer existed, it would try to find an alternative path from c (its current location) and its current goal (node E). Hence an alternative strategy could be ${}_hS_{SeqStart}a, b, c, x, E_{ANDStart}f, g, h, I_{ANDStart}j, k, l, M(50)$. The only difference is that the agent may move to host E via node x. Notice how only the path from c to E needed to be computed while the remainder of the complex strategy was reused. If an error occurred while the agent was executing parts of the complex strategy further down, say half way inside the first $ANDStart$ at node g then its target host would be node I. In this case, sections of the complex strategy on both the left and right sides of the point of failure (g to I) would be reused.

4 Some Related Work and Conclusion

Dynamic itineraries [3] consist of a minimum of at least one place to visit with subsequent places to visit being generated at run-time by the agent. Our approach is similar yet uses the notion of $goals$ to support dynamic itineraries. The strategies which are used to perform the mobile agent's actions according to [3] are only the sequential and parallel operations. This appears limiting to the agent when compared to ITAG's operators which also supports *Independent Non-determinism* and *Conditional Non-determinism*.

Having a flexible itinerary that can be altered at run-time ensures greater fault tolerance and is also referred to as *Hybrid Scheduling* [2]. In the hybrid scheduling approach, the agent re-computes a path for all of its goals, whereas ITAG-Goal only re-computes 'N' number of goal sets. Hence, in a highly dynamic environment the hybrid scheduling approach would re-compute its entire itinerary only to find that the network had changed once again soon after and would therefore have to re-compute all of its itinerary again until the agent had completed its tasks.

The notion of a more flexible itinerary can also be seen through the introduction of a `reconfigurable itinerary`, which is present in the JAMES mobile agent platform [5]. This system does not use the concept of goals like ITAG-Goal and concentrates more on the system-level where code recovery and re-start can take place.

We have introduced a two-level Goal-Strategy architecture for achieving flexible and robust mobility behaviour for agents. We have implemented agents with the ITAG-Goal interpreter and ran them against a simulated changing network. Our results demonstrated that ITAG-Goal agents can accomplish their itinerant goal, while agents with an equivalent fixed itinerary fails. Due to space limitations, we cannot reproduce details of the experiment in this paper. In our experiments, we resource-bound the agents and simulated how the agents utilize these resources each time they traverse a link or perform restrategizing on a node. There were a few cases where the ITAG-Goal agents failed, when they ran out of resources. Hence, though we have introduced greater flexibility through goals, this flexibility is limited by the amount of resources that the agent possesses. Whenever an agent restrategizes and attempts an alternative path, it utilises resources (whether time or fees for CPU time). There is a cost to being flexible, a notion we term *bounded flexibility*. In future work, we will reason about how committed the agent should be to a goal in the light of its resources and the state of the network.

References

[1] Franz Brandenburg. The generalized shortest path problem, 2002.

[2] Jau-Shien Chang; Chih-Yuan Chang. A visual mobile agent system with itinerary scheduling. In *Proceedings of the 4th International Conference on Autonomous Agents*, Barcelona, Spain, June 20-23 2000.

[3] Rahul Jha and Sridhar Iyer. Performance evaluation of mobile agents for e-commerce applications. In *Proceedings of the Eighth International Conference on High Performance Computing (HiPC 2001)*, December 2001.

[4] Christian Erfurth; Peter Braun; Wilhelm Rossak. Migration intelligence for mobile agents. Technical report, Computer Science Department, Friedrich Schiller University Jena; 07740 Jena, Germany, 2001.

[5] L.M. Silva; V. Batista; J.G. Silva. Fault-tolerant execution of mobile agents. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 135–143, New York, USA, June 25-28 2000.

[6] Michael Winikoff; Lin Padgham; James Harland; John Thangarajah. Declarative & procedural goals in intelligent agent systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, Toulouse, France, April 22-25 2002.

[7] S.W. Loke; A. Zaslavsky; B. Yap and J. Fonseka. Scripting mobile agents to support cooperative work in the 21st century. In *Proceedings of the Workshop on Agent-Supported Cooperative Work at Autonomous Agents 2001 (ASCW-2001)*, pages 1–10, Montreal, Canada, May 2001.