REVIEWS

Non-deterministic planning methods for automated web service composition

George Markou*, Ioannis Refanidis

University of Macedonia, Greece

Received: June 6, 2015 **Accepted:** August 26, 2015 **Online Published:** September 29, 2015
DOI: 10.5430/air.v5n1p14 **URL:** http://dx.doi.org/10.5430/air.v5n1p14

ABSTRACT

Web service composition (WSC) is the task of generating new composite web services that exhibit functionalities not supported by any single web service. In its simplest form this is achieved by linking existing web services in sequence. More complex forms link services in parallel or use alternative paths. WSC can be considered a planning task, with the web services being the planning operators and the initial state and the goals being provided by the user. Particularly, since web services operate in a stochastic environment, their output is not predictable, and the problem is formulated as a non-deterministic planning one. This article presents a critical, comprehensive and up-to-date review of the literature concerning alternative non-deterministic planning methods, including probabilistic planning, determinization methods, planning in the belief state space and translation-based methods. Furthermore, the article reviews existing implementations of WSC systems, employing a variety of planning approaches, and discusses the degree in which the current achievements from the non-deterministic planning field have been adopted successfully. To the best of our knowledge, this is the first review of its kind, one that provides a thorough introduction to the vast area of automated web service composition.

Key Words: Non-determinism, Planning, Web services, Composition

1. INTRODUCTION

Automated web service composition is the task of generating new composite web services that exhibit functionalities not supported by any single web service. Semantic web services can be efficiently translated to planning operators and the problem of composing a new web service using existing ones can be formulated as a planning problem, with the initial state being the current state defined by the user and the goal being the desired result, *e.g.*, having bought an item. Then, existing planning methods can be utilized to solve the resulting planning problem.

We assume that the outcome of the execution of a web service cannot always be anticipated. For example, using a web service to buy a book from a specific online bookstore may result in finding the book available, thus the user could proceed with buying it; alternatively, the book may be not available, so another web service should be used to look for the book in another online bookstore and, if available, buy it. Manually composing web services is a tedious task, due to the abundance of the available web services and the computational complexity of the planning task. Thus, employing efficient automated planning methods, particularly those that are able to tackle non-determinism, is desirable.

This article aims at providing an in-depth review of the research area of automated web service composition, giving particular emphasis to the non-deterministic nature of the

* **Correspondence:** George Markou; Email: gmarkou@uom.gr; Address: University of Macedonia, Greece.

problem. To this end, the article comprises an up-to-date survey of non-deterministic planning methods, focusing on contingent planning, the adoption of non-determinism in their formalisms, as well as a critical evaluation. Furthermore, the article demonstrates the utilization of planning techniques in state-of-the-art Web Service Composition (WSC) methods, through a survey of such recent applications, mainly those that incorporate non-determinism partially or fully. The article is structured as follows: Section 2 defines the theoretical background on non-deterministic planning and presents the relative planning approaches: naïve non-deterministic methods; determinization ones; methods that search through the belief space; and state-of-the-art translation-based approaches. Section 3 is devoted to WSC methods, reviewing deterministic methods, methods that generate multiple solutions or incorporate alternative web services to them and, finally, methods based on non-deterministic and contingent planning. Section 4 concludes the article and identifies future challenges.

2. NON-DETERMINISTIC PLANNING METHODS

This section reviews non-deterministic planning methods in fully, partially and non-observable domains, focusing on contingent planning techniques.

2.1 Background

Planning is the problem of deciding which actions have to be executed next in order to achieve a goal.[1] These actions can have fully predictable (deterministic) or unpredictable (non-deterministic) effects, with or without a model over their occurrence probability. Furthermore, the world can be either fully, partially or not observable at all. In a deterministic fully observable case, the outcome of an action is fully predictable and results in a single state; if s represents the state before the execution of action a and s' the state after its execution, then the transition function $s' = f(s, \alpha)$. In a non-deterministic setting, though, the next state depends on which effect of the action actually occurred. If the model is probabilistic, then an action can have n possible mutually exclusive effects, $e_1, e_2, \cdots, e_n$, each with probability p_i of occurring, $0 < p_1, p_2, \cdots, p_n \leq 1$, so that $\sum_{i=1}^{n} p_i = 1$. As such, the resulting states can be computed by $P_a(s'|s)$, with $\sum_{i=1}^{n} P_a(s'|s) = 1$. A domain may allow for observations that provide feedback about the action results and form the beliefs in regard to the possible states that occur.

Particularly, a non-deterministic planning domain is a triple $D = (S, A, \gamma)$, where S is a finite set of states, A is a finite set of actions and $\gamma : S \times A \rightarrow 2^S$ is the state-transition function. A fully observable non-deterministic planning problem is a triple $P = (s_0, s_a, D)$, where $s_a \in S$ is the initial state and $s_a \subseteq S$ is the goal.[2] Given such a planning problem, by $s_\pi \subseteq S$ we define the set of states to which an action has been assigned by policy π, where a policy is a function $\pi : S_\pi \rightarrow A$. That is, $\forall s \in S_\pi : \exists a \in A$ such that $(s, a) \in \pi$, and given a state $s \in S_\pi$, the solution dictates that the action to be executed is $\pi(s)$. Finally, $S_\pi(s)$ denotes the set of states reachable from s using π. A probabilistic planning domain is defined by $D = (S, A, \gamma, Pr)$. S, A and γ are the same as in the definition of general non-deterministic domains, and the probability-transition function is defined as $Pr : S \times A \times S \rightarrow [0, 1]$. The set of all actions that can be applied to state s is defined as $A_D(s) = \{\alpha \in A : \gamma(s, a) \neq 0\}$. A planning problem is also defined in the same way as a general non-deterministic one.

By considering Fully Observable Probabilistic (FOP) domains, world states can be completely observed at runtime. Following the previous notations, solutions to FOP planning problems can be (total) policies $\pi : S \rightarrow A$, or partial functions from a set S_π to A, with $s_0 \in S_\pi$ and S_π closed under policy π; then for $s \in S_\pi$, the solution π dictates to apply action $\pi(s)$ to state s. If $a \in A_D(s)$ then $Pr(s, a, s')$ is the probability of reaching state s' if action a is applied to s. We denote the execution structure for π with $\sum_\pi$, and the set of the states in $\sum_\pi$ with V_π. For any two states s and $s' \in V_\pi$, if there is a path in $\sum_\pi$ from s to s', then s' is a π descendant of s in $\sum_\pi$. Then, if for a state s it holds that there is no π-descendant of it that satisfies the goal, then s is a dead-end.[3]

A policy π is closed w.r.t. a state s iff $S_\pi(s) \subseteq S_\pi$. If the goal state can be reached using π from all (π-reachable) states $s' \in S_\pi(s)$, then π is considered proper w.r.t. a state s. Iff π is both closed and proper w.r.t. the initial state s_0, then it is deemed a valid solution.[4] Valid solutions can be either strong, or strong cyclic; a policy π is acyclic if for all possible executions $\tau = s_0 s_1 s_2 \cdots$ of π from s_0, it holds that $s_i \neq s_j$, for any $i \neq j$.[5] Solutions to non-deterministic problems can be either weak - with a chance of success, strong - guaranteed to achieve the goal despite non-determinism, or strong cyclic - guaranteed to achieve the goal with iterative trial-and-error strategies.[6] An alternative definition of different type of solutions is in terms of probability of success; in this sense, weak plans reach the goal with a probability $p, 0 < p < 1$. Strong acyclic plans reach the goal with probability $p = 1$, in at most $n \leq |S|$ steps, assuming that each outcome of an action a from state s has a non-zero probability. Strong cyclic plans reach the goal with probability $p = 1$, allowing for infinite paths but with probability $p = 0$ of reaching the goal.

A partially observable domain with sensing actions (PPOS)[7–9] is a tuple $D =< A, O, I, G >$. A represents the set of actions and O the set of observations; I is a set of clauses that specifies the initial state, and G is a conjunction of atoms specifying the goal. Literal l holds in a state s iff s assigns l to be true. The set of all states where l holds comprise the belief state b. For $a \in A$, its preconditions $Pre(a)$ are a conjunction of atoms, and its conditional effects $Eff(a)$ are a set of pairs $< s, l >$, where c is a condition and l is a literal, implying that l occurs in the next state if c holds in the previous state. An action a is applicable in a state s iff $Pre(a)$ holds in s; it is applicable in a belief state b iff a is applicable in every $s \in b$. In a similar manner, when an action a is performed in b, a successor belief state b' is constructed by performing a in each $s \in b$. Observations $o \in O$ are also represented by pairs $< c, l >$; when c is true, o denotes the truth value of l. This result is achieved by treating observations as "special" actions that have c as the precondition and l as the effect; thus, when such an observation action o is executed in b, the successor belief state b' is the maximal set of states in b agreeing on l. Moreover, a belief state b' is reachable from b if there is a sequence of actions and/or observations that when executed in b result in b'.

Contingent planning produces plans that are conditional, containing different branches for different observation action results, and can be constructed either in an offline or online fashion. The former allows for the generation of solutions with decision points based on the outcomes of observation actions. The resulting plans are larger but are more general and have the ability to avoid dead-ends.[7] However, the size of full tree-shaped contingent plans is exponential in the maximum number of observations contained in a single branch of the tree.[10] Offline methods can output both belief policies and tree ones. A belief policy is a function mapping belief states into actions, whereas a tree policy is a function mapping executions into actions,[10] with the former being represented by graphs and the latter by trees.[11] Online methods focus on choosing the appropriate action for execution for the current belief state.

2.2 Probabilistic planning

One of the first methods that used classical planning techniques to tackle probabilistic problems was Mahinur,[12] a probabilistic partial-order planner that generates a (weak) base plan and searches for the contingencies, which, if they were to fail, would affect the plan the most; it then improves the plan based on them using preventive and corrective repairs, as well as replacement of branching actions. However, attempts such as Mahinur or Cassandra[13] are not competitive with modern planners.

APROPOS2 is an anytime probabilistic contingent planner that chooses the most likely outcomes of all actions;[14] it then generates a base plan under this assumption and considers less likely outcomes to modify it. APROPOS2 is based on a conversion of the original probabilistic problem to a satisfibility (SAT) one and generates solutions for partially observable (probabilistic) propositional problems.

Meuleau and Smith[15] present a method for fully-observable automated bounded branching planning.[16] They define three variants of k-contingency planning, with the plan form that contains k branches being the differentiating factor. An anytime, optimal, balanced k-contingency planning method is chosen, with plans having at most k branch points in each trajectory that leads to the goal based on Partially Observable Markov Decision Processes (POMDPs), to compute plans for non-deterministic problems with full observability.[17] For $k = 0$, the output is an action sequence solving a conformant problem; for $k = \infty$, the algorithm outputs the optimal policy. The method was evaluated in two domains taken from Kaelbling *et al.*[18] and Hyafil and Bacchus.[19]

2.3 Determinization methods

One of the prevalent methods to tackle non-deterministic problems is the determinization of the original non-deterministic domain and the generation of a solution to the deterministic problem that is also a weak solution of the original problem. Table 1 summarizes the methods that are presented in this section, along with some of their features.

FF-Replan[22,23] utilizes the FF planner to generate a single plan for a deterministic version of the original Markov Decision Process (MDP) problem, whereas it produces a new plan each time one of its simulated executions of the current plan results in an unexpected state (from that state to the goals). Yoon *et al.*[20] introduced two methods of creating a determinized version of the original non-deterministic problem. The first was single-outcome determinization, which selects a single effect for each non-deterministic action, without taking into account the rest of its effects or its probabilities (if any). The second one, all-outcomes determinization, creates a new action for each non-deterministic effect of the original action. Assuming a non-deterministic planning domain D with a set of actions $A = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, then for an action $\alpha_i \in A$ with k different non-deterministic outcomes, the all-outcomes determinization is the set of deterministic actions $Det_{\alpha_i} = \{\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{ik}\}$ along with a state-transition function $\overline{\gamma}$ such that for every states, $\gamma(s, \alpha_i) = \overline{\gamma}(s, \alpha_{i1}) \cup \cdots \cup \overline{\gamma}(s, \alpha_{ik})$;[24] the single-outcome determinization for action α_i simply chooses one of the actions Det_{α_i} to substitute α_i.

Table 1. Synopsis of determinization methods

Method	Online/Offline	Complete	Replanning	Determinization	Plan strength
FF-Replan	Online	-	•	All-outcomes Single-outcome	Weak
NDP	Offline	•*	-	All-outcomes	Strong cyclic
RFF	Offline	-	•	Single-outcome	Weak + robust‡
FF-Hindsight	Online	-	-	Hindsight	Weak + robust‡
Jiménez *et al.*[29]	Online	-	•	All-outcomes + costs	Weak + robust ‡
Foss *et al.*[30]	Online	-	•	All-outcomes + costs	Weak + robust ‡
Dearden *et al.*[31]	Offline	-	-	Single-outcome	Weak + robust ‡
GPT	Offline	•#	-	Single-outcome dynamic†	Weak + cost optimal

* Under conditions
# For certain classes of problems
† Multiple single outcome determinizations/trials are used
‡ The output plans are weak but with some guarantees against failing / have been made more robust against contingencies

The all-outcomes determinization process is also used by FIP (Fast Incremental Planner)[2] and by NDP.[24] In contrast to FF-Replan, though, both NDP and FIP generate strong cyclic solutions. NDP is very similar to FF-Replan, but it may also be complete provided that a complete planner is employed. The all-outcomes determinization, however, ignores the probabilities attached to the probabilistic outcomes. The planners utilizing it can simply choose the most convenient action, one that achieves the desired effect, regardless of its underlying (true) likelihood. As such, even actions with trivial probability of succeeding can be selected if they generate the most helpful outcome. Despite this fact and that it does not offer any quality guarantees, FF-Replan was very successful in planning competitions and was used as the basis for various non-deterministic planning methods, *e.g.*, Ref.[25]

In contrast to FF-Replan, RFF generates offline partial policies and provides some guarantees against failing. Multiple plans are generated using single-outcome - the most probable one - determinization, and the policy that is produced has a low probability of causing replanning during execution through Monte-Carlo simulation. If the failure probability is sufficiently low, the current policy is returned; otherwise, it is further expanded. If the execution results in a state for which the policy does not specify any action, RFF tries to reach neighbor states for which an action has already been specified, instead of replanning. As a last resort, replanning may be required, though, as RFF is not optimal and it does not handle dead end states.

Yoon *et al.*[26] present FF-Hindsight, a generalization of FF-Replan, which randomly produces multiple determinized problems and combines their solutions. The value of each state is approximated by sampling these (non-stationary) problems originating from i; the problems are then solved in hindsight and the values of the states are combined. In that way, the determinization process is no longer static as in FF-Replan, which can be considered an optimistic approximation of hindsight optimization. The amount of computation required at each state is linear in the number of actions applicable in it, as well as in the number of determinized problems generated.[27] For this reason, in Yoon *et al.*[27] several improvements to FF-Hindsight are incorporated in FF-Hindsight+, *e.g.*, the detection of potentially useful actions, the reuse of relevant plans and the modification of the action evaluation method, utilizing the all-outcomes determinization instead of the sampled one. FF-Hindsight+ proved to be competitive with past winners of the International Probabilistic Planning Competition, namely FF-Replan, FPG[28] and RFF (respectively for IPPC 2004, 2006 and 2008).

Jiménez *et al.*[29] combined the all-outcomes determinization with a translation of the outcomes' probabilities to associated cost values for the new deterministic actions' outcomes, each denoting the risk of failing. In that way, a metric planner compliant with PDDL can improve the plans' robustness by trying to minimize the sum of the negative logarithms of the success probabilities, which in turn minimizes the product of the failure probabilities. Foss *et al.*[30] use a determinization translation process for incremental contingency planning; an initial deterministic seed plan is generated that is then iteratively improved through an analysis and repair cycle in order to find outcomes that are relatively probable and result in dead ends; then, precautionary actions are added in relation to the most problematic outcomes. Recoverable failures, on the other hand, are left as-is in the plan and are repaired through online replanning.

Dearden *et al.*[31] formulate the domain differently; the initial state may involve uncertainty about the variables that participate in it, characterized by probability distributions

over their different values. The same is true for the actions' probabilistic effects. A seed plan is initially constructed though the use of a deterministic planner, assuming that each action executes as dictated by its expected behavior. Then, additional branches are generated and added to the existing plan incrementally, so as to improve its overall utility. The best place to insert a branch is computed based on the utility gained from adding it at the specific place. As this computation is expensive, an approximation is used through Monte Carlo simulation and the propagation of utility distributions through a graph. This process is repeated until the plan is sufficiently robust, or it runs out of time.

Bonet and Geffner[32] formulate the problem of non-deterministic planning in partially observable domains as a search in belief space, with each belief state representing a set of states or probability distributions over states. The proposed method, GPT (General Planning Tool), uses the max heuristic to guide the search.[35] It is an admissible heuristic that measures the positive interaction in the belief states and provides an estimate of how difficult it is to achieve the subgoals in the worst case. GPT makes use of a generalization of the LRTA*[33] algorithm, named RTDP (Real Time Dynamic Programming),[34] to construct a policy by using multiple simulations of an execution from an initial state to a goal one, based on the current approximation of the final policy. After each action is selected/executed, the policy is updated.

A variant of GPT, mGPT,[36] extracts different classes of lower bounds by determinizing the original problems and uses them in combination with various heuristic-search algorithms that use these lower bounds to focus their policy updates.

2.4 Planning in the belief state space

This section presents a set of planners that tackle the problem of non-deterministic planning in belief state space so as to scale up efficiently. Table 2 provides a synopsis of the reviewed methods and their features.

Table 2. Synopsis of methods for planning in the belief state space

Method	Full / Partial / No Observability	Scale up method	Plans
MBP	•/•/•	BDDs	Strong/ Strong cyclic†
YKA	•/•/-	BDDs	Strong
POND	-/•/•	LUG	Strong
PC-SHOP	-/•/-	Domain-specific knowledge	-
ND-SHOP2	•/-/-	Domain-specific knowledge	Strong cyclic
YoYO	•/-/-	Domain-specific knowledge + BDDs	Strong / Strong cyclic
NDP2	•/-/-	Abstraction + All-outcomes determinization	Strong cyclic
Gamer	•/-/-	Translation to two-player turn-taking game + BDDs	Strong / Strong cyclic
Fu *et al.*[58]	•/-/-	MRDAG + Heuristics	Strong
FIP	•/-/-	State re-use + goal alternative	Strong cyclic
PRP	•/-/-	State relevance + All-outcomes determinization	Strong cyclic
Prob-PRP	•/-/-	Avoidance of state-action pairs leading to dead-ends	Strong cyclic‡
Winterer *et al.*[63]	•/-/-	Stubborn sets + FIP / LAO*	Strong cyclic
Contingent-FF	-/•/-	Implicit representation of belief states	Strong
FPG	-/•/-	Local optimization + function approximator	-
DNFct	-/•/•	DNF Representation of belief states	-

† Depending on the underlying algorithm, under full observability, MBP may generate weak, strong or strong cyclic plans. Under partial or no observability, the plans are strong

‡ In domains where all dead-ends are avoidable

Model Based Planner (MBP)[37] is a general non-deterministic planner that deals with domains that incorporate different degrees of observability: fully, partially observable, or not observable at all. It supports uncertainty in the initial state and the actions, as well as temporally extended goals. Domains are considered to be non-deterministic finite-state machines (FSMs) and plans are defined as deterministic FSMs. The plans are generated with the help of Symbolic Model Checking techniques, with belief states representing sets of states that contain common observations; they are defined as propositional formulae and the state space search comprises logical transformations over them. The use of Bi-

nary Decision Diagrams (BDDs) allows not to enumerate the state space explicitly and to ignore irrelevant information.[38] However, their size is sensitive to variables' order and may be very large, while the computation of the belief states' successors requires the generation of intermediate formulae that may be of exponential size.[39] Thus, the methods utilizing them may suffer from an explosion in state space and not scale well.[6]

MBP utilizes various planning algorithms to solve the problem, and depending on its type, generates plans of different form and guarantees of reaching the goals. In the case of partial or no observability, the generated plans are strong. Cimatti *et al.*[6] compare MBP to various other planners, namely GPT, UMOP,[40] SIMPLAN,[41] QBFPLAN[42] and SGP.[43] MBP proved effective in a wide range of problems, although a blow up in its state space occurred even in problems with a limited number of state variables. MBP clearly outperformed only UMOP, as its comparison with the rest of the planners indicates that each suffers from a different drawback, *e.g.*, SGP's bottleneck stems from the enumeration of the initial states. However, the systems under evaluation solve different problems, formulated in different languages, with each problem encoding having an impact on the planners' performance.

BDDs are also used by Rintanen[44] who presents two backward search algorithms and argues that forward search algorithms are required to choose between branching and performing an action; the former leading to larger plans, the latter exchanges useful branch points for potentially smaller plans. The steps of these algorithms represent both the application of actions and branching in the plan. In the case of partial observability, the first algorithm exhaustively compute sets containing all the maximal belief states with an increasing distance to the goal (belief) state, with the distance representing the maximum number of actions needed to reach the goal from the belief states in the set. The second one heuristically selects a single belief state at a time based on its cardinality. Both algorithms are suboptimal and generate solutions in the form of directed acyclic graphs (DAGs). In the case of full observability, the algorithms perform breadth-first search backwards from the goal states, traversing the entire search tree up to a level.

In this case the plan search is essentially a computation of the distance from every belief state to the goal; based on these distances it is straightforward to extract a plan. This blind enumeration of the belief state space, however, proved to be inefficient, as their high number becomes a bottleneck to the problem's solutions. This led to a new factored representation of the belief space that is able to identify new belief states without blind enumeration, while also allowing algorithms to generate non-deterministic plans backwards.[45] In this way, all branching points can be automatically handled, and in the case of fully observable planning problems, the representation allows for a single BDD to represent the entire belief space. This new method, YKA, is competitive with MBP, solving all the problems in the evaluation. MBP is, however, considerably faster in a number of problem instances.

Bryce *et al.*[46] present POND (Partially-Observable Non-Deterministic planner), a planner with the main advantage of using a single Labeled Uncertainty Graph (LUG) to plan in belief space. LUG labels the propositions/actions with formulas that describe the initial worlds in which they can be applied;[47] it also guides the search through providing a heuristic for a progression (in the contingent setting) planner. LUG condenses the information in the states comprising a belief state in a single graph representing their optimistic projection and is used to estimate the number of actions required to reach each belief state. Its generation ends when the goal (belief state) can be achieved by the literals appearing in one of its levels, provided these literals have labels that indicate that the goal is supported in all possible worlds. The fact that the goal must be supported in all possible worlds is similar to the max heuristic of GPT; the one used in POND, however, is inadmissible. The search in the state space is performed with a top down AO*[32,48] method, in which the nodes are belief states and the hyper-edges are actions.

Bryce *et al.*[49] compare heuristics based on various graph data structures. They conclude that heuristics that do not use graphs, or use a single one, have limited ability to help planners scale, while ones using multiple graphs scale better, but are computationally costly. Multi graph heuristics are deemed better than single graph, with LUG providing the best results in terms of scalability. The evaluation examined the performance of POND against MBP, GPT, SGP and YKA, with the heuristics used in POND being more effective and informative in comparison to cardinality heuristics and the max heuristic in GPT. The evaluation also indicates that POND is up to par with BDD-based planners (MBP and YKA) and Graphplan-based ones (SGP).

PC-SHOP[50] is a method based on the classical Hierarchical Network Planner (HTN) planner SHOP.[51] SHOP was extended to handle partial observability and probabilistic effects, with uncertainty in the problems modeled using explicit enumeration of belief states. Similar to SHOP, PC-SHOP makes use of domain-knowledge to solve the problem at hand, and being a depth-first search algorithm it makes use of an iterative deepening strategy to prune deep recursive calls of methods. The generated solutions have a tree-structure and

no merging of their branches is attempted. ND-SHOP2[52] is based on the successor of SHOP, SHOP2,[53] and extends it to address fully observable domains with non-deterministic actions and multiple initial states. The search is based on a forward-chaining algorithm that exploits domain-specific search-control heuristics. In case these techniques allow for a significant pruning of the search space, ND-SHOP2 outperforms MBP. In the general case, since the planner does not use symbolic representations of belief states, it scales worse. For this reason, YoYO[54] combines the HTN-based search of ND-SHOP2 with BDD-based symbolic model checking. In that way, it does not enumerate belief states explicitly as in PC-SHOP or ND-SHOP2, outperforming both methods in all cases, while also solving larger problems.

Alford *et al.*[55] presented NDP2, a method improving the NDP algorithm that concerns its behavior towards unsolvable states. NDP2 uses the all-outcomes determinization to produce multiple weak plans and combines them in an effort to build a strong cyclic solution. In order to deal with unsolvable states, NDP2 modifies the resulting deterministic problem by rendering some of its actions inapplicable at the first step of any solution, so as to find acyclic plans that avoid visiting known unsolvable states. This results in a quadratic increase of the domain description per constrained action, instead of an exponential amount in NDP. NDP2, using FF, was compared experimentally to MBP; the results do not indicate that one clearly outperforms the other. In most cases, the abstraction mechanisms of NDP2 are significantly less efficient than the use of BDDs, but, the use of an external planner allows NDP2 to visit fewer states than MBP.

As mentioned in Section 2.3, another method that attempts to improve on NDP is FIP; FIP iteratively expands a graph of the reachable states until a path to the goal exists for every (non-goal) leaf state.[2] That is, an external classical planner, in this case FF, generates a weak plan for an arbitrarily selected non-goal leaf state. If the classical planner used is complete, then FIP is also complete. Furthermore, FIP was extended, first, by keeping track of the search results of each iteration, allowing it to avoid exploring the same (solved) states more than once and, second, by the addition of an alternative goal heuristic. For each action with non-deterministic effects, the effect included in the current weak plan represents the intended one; the other one represents a failed effect. Then, instead of setting the goal for a weak plan to be the original goal, the heuristic searches for a weak plan that leads to the intended effect. If one is not found, search is restarted to find a weak plan for the original goal.

FIP was evaluated against MBP and Gamer; Gamer,[56, 57] the winner of the Fully Observable Non-Deterministic (FOND) track at the 2008 IPC,[4] presented a novel method to solve non-deterministic problems, by translating the original problem into a two-player turn-taking game. The translation, which is linear in the parameterized domain, formulates the problem as one in which one player represents the desired moves dictated by the planner and the other player represents the non-deterministic effects in the environment, by compiling each non-deterministic action into two. Gamer solves the problems optimally, producing strong or strong cyclic plans,[6] and outputting a state-action table in the form of a BDD. Both versions of FIP, with and without the extensions, have the same problem coverage, outperforming MBP and Gamer significantly. The extensions are particularly beneficial to FIP, as the planner is considerably faster and produces plan sizes that are significantly smaller.

Fu *et al.*[58, 59] propose a FOND planner that produces strong plans based on a multi-root directed acyclic graphs heuristic (MRDAGs); the MRDAGs are used to define the expansion of the search space by distinguishing between states with one or more actions. This is important as backtracking is essential in strong planning in order to avoid cycles, and whenever one is encountered, backtracking continues until it reaches a state with more than one applicable actions. The heuristics, on the other hand, are utilized so as to choose the order of the applicable actions in a state, if more than one are available. Two heuristics are proposed; the first heuristic, MCS, sorts the states in increasing order of their number of applicable actions. The second one, LHD, sorts the states in increasing order of their heuristic distance to the goal, which is calculated based on the FF heuristic.[23]

The proposed method was evaluated against Gamer and MBP, on problems from the 2008 IPC.[4] Gamer outperforms MBP considerably, however the approach in Fu *et al.*[58] is orders of magnitude faster than both, exhibits significantly better scalability, and produces solutions of approximately the same length. The results also indicate that LHD is far more beneficial than MCS, *i.e.*, the order in which the states are expanded in an MRDAG is not crucial to planning efficiency.

Muise *et al.*[60] presented PRP (Planner for Relevant Policies), a planner that adopts the basic idea behind FIP and combine it with regression search. PRP converts a non-deterministic domain to a non-deterministic SAS+ formalism and returns a strong cyclic plan, or the best policy, if the former does not exist. It uses the all-outcomes determinization to search for a weak plan each time a state that is not handled by the current policy is encountered; only the relevant parts of the states are used during search, and the determinized problem is solved for various initial states. A weak plan is generated, and when a state that does not have a strong

cyclic plan is encountered, replanning is employed. PRP was extended by using local planning, *i.e.*, in case an unhandled state is encountered, PRP generates a local plan to get back to the intended state instead of replanning. Furthermore, it identifies states in which the policy essentially acts as a strong cyclic one, instead of exhaustively enumerating them. The first extension, however, is efficient only when the local plan is extremely short.

PRP was implemented by modifying Fast Downward (FD);[61] as an offline FOND planner, it significantly outperforms FIP, in relation to the output plan size and the planner's run time and manages to avoid potential dead-ends through simulation. It achieves the goal with up to several orders of magnitude fewer actions than FF-Replan, and when the problems are "probabilistically interesting"[62] with possible dead-ends, PRP also scales better than FF-Hindsight+ and is significantly faster. Although counterintuitive, as in domains with probabilistic action outcomes PRP simply ignores the probabilities attached to the actions and behaves as a FOND algorithm, PRP manages to solve most benchmark problems.

PRP was extended in Camacho *et al.*[63] to solve probabilistic planning problems offline, maximizing the probability of reaching the goal, while also aiming to maintain a balance between the expected plan length and their compactness. The proposed method, Prob-PRP, generates strong cyclic plans in domains where all dead-ends are avoidable. To tackle such domains, it identifies state-action pairs that may lead to undesired states, thereby efficiently pruning the search space. In case a solution is returned with a probability of reaching the goal less than 1, the problem at hand contains unavoidable dead-ends. Prob-PRP was evaluated against RFF and was shown to scale up better, require even orders of magnitude less planning time than RFF, and produce solutions of better policy in domains with avoidable dead-ends. In problems with unavoidable dead-ends, although better than RFF, Prob-PRP struggles, occasionally exceeding the set memory limits or not converging within the time limits.

Winterer *et al.*[64] directly utilize FIP in combination with deterministic stubborn sets[65] and greedy best-first search with the FF heuristic[23] as the underlying classical planner. They also suggest a different method, that of using LAO*[66] with a non-deterministic stubborn set formalism. Both methods were shown to have better performance than FIP, solving more problem instances and expanding fewer nodes; in the latter case, however, the expanded nodes were not as drastically reduced.

Hoffmann and Brafman[47] present Contingent-FF, which treats contingent planning as search through an AND-OR tree in the space of belief states. Contingent-FF represents belief states implicitly through the action-observation sequences that lead to them from the initial state. During search, the propositions that are known in each belief state are computed, with a proposition being considered known in a belief state if it holds in the intersection of the worlds in that belief state. As such, Contingent-FF scales up efficiently and does not require large amounts of memory,[39] but it has to reason about the entire action sequence that lead to a belief state.

Search is performed as a weighted AO* forward one, with OR nodes representing belief states and AND nodes representing actions. It is assumed that actions with unsatisfied preconditions do not cause the plan to fail, but simply do not produce any result. A relaxed problem is produced by ignoring the delete lists and the length of the conformant solution for it is used as the heuristic. The method was experimentally compared against POND and MBP, that is, with methods that tackle the way belief states are handled in different manners. The experiments show that the plans generated by Contingent-FF were, in most cases, similar in terms of quality to those generated by POND and better than those generated by MBP.[47]

Buffet and Aberdeen[28] present the Factored Policy-Gradient planner (FPG), a probabilistic temporal planner aimed to efficiently tackle large partially observable problems using Reinforcement Learning and a local optimization procedure, *i.e.*, online gradient ascent, to search for plans. Gradient ascent is used for direct policy search, by estimating the gradient of the long term value of the process. The gradients represent the output policy, which is then factored into simple approximate policies for starting each action. These policies map a partial observation to the actual probability of executing the action, representing the amount of usefulness of each one. FPG also aggregates similar states in an implicit manner, by having each policy contain critical observations and not entire states; moreover, Monte-Carlo-like algorithms are used to keep the memory consumption independent of the state space size.

FPG can optimize both the makespan and the probability of reaching the goal, but the techniques used can lead to suboptimal policies; moreover, FPG may be inefficient in large domains if it does not reach the goal in a short time, and it has long learning times since it improves upon an initially random policy. As such, an improvement of FPG, FF+FPG,[67] guides its search of the state space by using FF's heuristic, specifically by having FF return the action it would execute in a given state and caching them for efficiency reasons. In that way, FF+FPG follows a policy based on FF while trying to learn its own, using importance sampling. In

general, FF+FPG manages to learn better policies than FPG, solving problems that FPG could not solve.

Finally, To *et al.*[39] utilize an earlier idea from To *et al.*[68] to encode the belief space in a compact form of disjunctive formulae, called minimal DNF; the resulting planner, DNFct, under certain restrictions, allows for the computation of belief states' successors in polynomial time. The transition function is modified so that it can also handle non-deterministic action effects and observation actions, making the method suitable for PPOS problems. The underlying algorithm used to generate solutions for these problems is an AND/OR forward search one, called PrAO (Pruning AND/OR search). PrAO is based on standard AND/OR graph search algorithms; it is extended so as to prune useless nodes and scale up more efficiently, and keeps track of potential solutions, allowing the remaining AND/OR search graph to comprise the solution tree for the problem, when the search terminates.

2.5 Translation methods

This section presents a set of methods that are based on the translation of the original non-deterministic problem into another one; this can be another form of non-deterministic domain that can be tackled more easily, *e.g.*, a FOND one, or a classical deterministic problem. These methods are similar in principle to the ones presented in Section 2.3; however, the main difference is that they utilize translations that are, in most cases, complete, so the resulting problems are equivalent to the original ones. Table 3 summarizes the methods presented in sections 2.5.1 and 2.5.2 and their features.

Table 3. Synopsis of translation methods

Method	Target Contingent Problem	Translation to	Complete	Translation Complexity
CLG	Simple	FOND problem	•	Polynomial
CLG+	With dead-ends and $width \geq 1$	FOND problem	•[#1]	-
K-Planner	$Width \geq 1$, with restrictions[#2]	FOND problem	•[#3]	Linear
LW1	Simple	FOND problem	•[#4]	Linear
PO-PRP	PPOS, with restrictions[#5]	FOND problem	•	Quadratic
SDR	PPOS with deterministic actions	Classical problem	•[#6]	-
MPSR	PPOS with deterministic actions & observations	Classical problem	•	Double exponential
HCP	PPOS with deterministic actions & observations	Classical problem	-	Linear
Palacios, *et al.*[10]	PPOS with deterministic actions & observations	Classical problem	•	Polynomial

[#1] For suitable choices of tags and merges
[#2] Problems with invariant non-unary clauses representing the initial situation's uncertainty and hidden fluents in the initial state that do not appear in the body of conditional effects
[#3] On the condition that the problem's state space is connected
[#4] For simple contingent problems
[#5] Problems whose initial state comprises only state invariants and in which uncertainty decreases monotonically
[#6] If the underlying classical planner is sound and complete

2.5.1 *Translation to FOND problems*

A contingent problem P involving uncertainty only in the initial situation and comprising only deterministic actions, cannot be translated to an equivalent classical problem P', since the two problems have different solution forms.[69] However, contingent problems can be translated to equivalent FOND problems $X(P)$ in state space, which, for strong solutions, have a similar solution form. Such a translation is presented in Albore *et al.*,[69] in which the non-deterministic actions in $X(P)$ represent the sensing actions of the original problem.

In that way the computations can be made in regard to states that are represented by sets of literals instead of sets of states, which is computationally expensive. This translation can be complete, as well as efficient; it can also be polynomial in time if the problem has bounded contingent width. Problems with a bounded contingent width of 1 are simple, and Albore *et al.* show that almost all the existing benchmark planning problems are such. Simple contingent problems are characterized by the fact that the uncertainty is related to the initial values of a set of multi-valued variables that are used in goals or action preconditions, but it is not present in the body of conditional effects.[9] Moreover, in simple problems, the value of any proposition that appears within an effect condition is initially known and constraints on the value of initially unknown propositions are invariant.[70] An example of such a problem is the well-known Wumpus,[71] where an agent can move in a grid orthogonally with the goal of reaching a destination containing a treasure. The agent has to avoid a monster named Wumpus that lies in an unknown

location inside the grid and can do so by detecting the Wumpus if he is in an adjacent location from its smell. In this case, the locations that contain a Wumpus are the invariant hidden variables, as they are initially unknown, never change and do not affect other variables.

Even $X(P)$, however, cannot be solved trivially. As such, an external classical planner is used on a relaxation of $(P), X^+(P)$,where all the deletes, preconditions, and actions with non-deterministic effects of $X(P)$ are dropped. Its solution, which can be obtained in polynomial time, is used as an estimate of the size of the plans of $X(P)$. A variant of $X^+(P)$ is used, with the resulting (classical) problem being called the heuristic model $H(P)$. A special case of the general translation scheme is presented, $X_i(P)$, which is complete and polynomial in the fixed factor i, if the contingent width of the original problem P is less than i. The presented Closed-Loop Greedy (CLG) planner uses the $X_1(P)$ translation, therefore, it is complete and polynomial only for simple contingent problems. $H(P)$ is used to select the next actions in a closed-loop fashion, starting from the initial state s being equal to $X_1(P)$, generating an action sequence to be applied in s that results in s' through the use of a modified version of FF, continuing in this fashion until s' is a goal state.

CLG can be used in an online or offline fashion; the result of its online use is a single successful execution, while in offline mode it generates full contingent plans. CLG was evaluated against Contingent-FF and Pond 2.2. In online mode it can solve larger problems; in offline mode it outperformed both planners. Another comparison between CLG, Contingent-FF and POND 2.2 exists in To *et al.*,[39] where the aforementioned planners were evaluated against DNFct; DNFct was the best planner overall, with CLG being second best and the other planners demonstrating poor performance.

In contingent planning, a belief state in which a dead-end state is reachable is itself a dead-end, as the planner cannot reach the goal with certainty from it. Despite its success, CLG could not perform well with dead-ends and was subsequently extended so as to work in contingent problems without solutions.[72] Among other modifications, assumptions that can be confirmed or refuted based on the gathered observations were incorporated into the planning process, and the classical planner used, FF, utilized a different heuristic, namely the set-additive one h^s_a.[73] The resulting planner, CLG+, can solve contingent problems with contingent width higher than 1, in contrast to CLG and Contingent-FF and more efficiently than POND.

Similarly to CLG+, K-Planner[9] extends CLG and is not restricted to simple contingent problems. It introduces a linear - instead of quadratic - translation scheme and requires that the values of the non-unary clauses representing the uncertainty about the initial situation are invariant throughout the execution of the plan. Moreover, it is assumed that the hidden fluents of the initial state do not appear in the body of conditional effects, which are assumed to be deterministic.[9] Such restrictions in the Wumpus domain would require that the Wumpus remains in the same position throughout the planner's execution. The presented translation is complete on the basis of the aforementioned assumptions, provided that the search space of the problem is connected. The solutions can be produced using a classical planner; sensing actions in K-Planner are translated into multiple deterministic actions that provide the knowledge of different values of the sensed variable,[74] with the planner being allowed to choose the value it will sense. With these assumptions, the method falls in the category of planning under optimism. K-planner was compared favorably against other approaches, *e.g.*, CLG in Shani *et al.*[75] However, as Shani and Brafman[74] note, the aforementioned assumptions allow K-planner to maintain belief states more easily and generate efficient translation patterns, thus providing it with a clear advantage against other more general planners.

A method that combines the completeness of CLG for simple, *i.e.*, of width 1, contingent problems, with the linear belief tracking translation of K-Planner is presented in Bonet and Geffner.[76] The presented planner, LW1 (Linear translations for Width-1 problems) is an on-line partially observable planner that generates a classical plan from the heuristic model $H(P)$ and executes it until the first sensing action. If the sensing action's true value is the same as expected, then the execution proceeds; otherwise, a new classical plan is computed from the current state. Width-1 completeness is achieved through explicitly defining originally implicit conditional effects, instead of using tagged literals as in CLG. This results in the method utilizing two linear translations; the first, for belief tracking, the second for action selection using classical planners. LW1, using FF as the underlying classical planner, is compared experimentally against K-Planner and the Heuristic Contingent Planner (HCP);[70,75] LW1 is the fastest planner in the comparison in most domains, and generally produces shorter execution plans; furthermore, it also scales up better than both planners, with HCP being the worst of the three in terms of scalability.

PRP, presented in Section 2.4, was subsequently extended to also tackle PPOS domains;[7] in specific, PO-PRP considers domains with incomplete information regarding the initial state and all the available actions are treated as deterministic, with the exception of the sensing ones. Similar to CLG+, PO-PRP assumes that the initial state comprises

a set of state invariants constraints. A further assumption is that the uncertainty in the domain cannot increase; once a previously unknown property becomes known, it cannot become unknown again. The compilation presented in Bonet and Geffner[9] was subsequently used in Muise *et al.*[7] to convert the PPOS problem at hand to a FOND one and then use PRP to compute strong cyclic plans that are subsequently converted into DAGs. PO-PRP was experimentally evaluated against CLG, which it outperformed in relation to both the size of the generated solutions and the required time to compute them.

2.5.2 *Translation to classical problems*

Brafman and Shani[77] tackle probabilistic domains with partial observability, sensing and deterministic actions; their planner, SDR (Sample, Determinize, Resample), generates a solution for a determinized version of the original domain based on the T0 translation,[78] which allows for the incorporation of the current knowledge of the agent in each state. Then, SDR follows the plan until the goal is met, unless the preconditions of the next action do not hold in all possible states, in which case a new plan is generated.

As in Albore *et al.*,[69] sensing actions are translated to non-deterministic actions; furthermore, the translation may lead to domains with an exponential (in the input size) number of propositions. In Shani and Brafman[74] it is proposed that both of these problems should be tackled through state sampling; an arbitrary possible initial state is chosen and the planner assumes that the observations will be sensed as if this is the actual initial state. If that proves not to be the case and the chosen initial state is not consistent with the world, replanning is employed. In regard to the representation of the belief states, SDR adopts an implicit method similar to Contingent-FF and simplifies it by only maintaining the actions and the observations made, along with the chosen initial state. In this way, only the history of execution is required to determine whether a condition holds. In relation to Contingent-FF, this is easier to compute and produces smaller formulas, despite being prone to reconstructing a formula – or parts of it – multiple times. SDR, using FF, was evaluated experimentally against the online version of CLG, being faster and scaling up better.

As aforementioned, CLG and K-Planner fall into the category of planning under optimism by allowing the planner to select the values that it senses. SDR samples a few initial states and chooses an arbitrary one from them, assuming that the observations will be sensed as if it is the true one and disregarding all other execution paths. Both methods are not realistic, and can also be ineffective, as until replanning encompasses the new information, a few actions may have already been executed based on the old assumptions of the world, possibly leading to dead-ends.

Brafman and Shani[79] presented a method designed to tackle such issues called MPSR (Multi-Path Sampling Replanner); MPSR is a complete translation method, the solutions of which comprise a contingent plan that takes into account all the possible execution paths. The main ideas behind it are the notion of distinguishability between states and the enhancement of the original set of actions. MPSR keeps track of which states are distinguishable by adding propositions to the domain that define whether two states are distinguishable.

The latter idea is used so that a contingent plan can be generated within a classical - linear - one; the original set of actions is modified with new ones, each of which has the following behavior: An action a_b is generated for an original action a that produces the same effects as a in states that are in belief state b, but does not produce any effect for any state that is not in b. As such, for different belief states, different versions of the original action are generated; this, however, leads to a significant increase of the number of actions and the size of the generated problem.

If used offline, MPSR utilizes a classical problem with its size being linear in the number of initial states and exponential in the number of propositions, and generates a complete contingent plan if one exists. However, a modified version of MPSR that uses an incomplete translation process is preferred; this version relies on replanning, similarly to CLG and K-Planner, and also makes use of sampling of a subset of the initial states, similarly to SDR. Moreover, the belief maintenance and update is also similar to SDR and Contingent-FF. At each iteration, a multi path translation generates a plan based on a subset of the initial states that is not a valid solution plan; it is, however, more informed than the plans generated by SDR, as it takes into consideration more than a single initial state. This plan is executed until either a new observation is made, altering the belief state and requiring the modification of the agent's knowledge, or until the next action cannot be executed due to its preconditions not being met. MPSR was evaluated against CLG and SDR, with all planners using FF as the underlying classical planner. MPSR proved faster than both planners, although in domains with dead-ends it generates longer plans. If the evaluated domain does not contain dead-ends MPSR generates smaller plans and scales better than CLG, with SDR not being able to solve it.

HCP is an online method for contingent planning that utilizes landmark-based heuristics[80] to select the next reachable sensing action. Landmarks are essentially used to identify the appropriate sensing actions, which are viewed as sub-

goals of the problem; in this way, the search space can be considerably reduced. HCP is greedy; it selects a single sensing action, after which classical planning is used on a modified version of the original problem; this process is repeated until the goal can be reached without requiring any more information. Also, in contrast to other planners, *e.g.* CLG, it does not require explicit modeling of belief states or translation techniques that may require exponential space.[70] HCP extends the translation scheme presented in Bonet and Geffner[9] in order to handle non-simple contingent domains.

The modified translation is not complete and, in contrast to K-Planner that plans directly over it, HCP utilizes it to obtain landmarks for the original problem and as a projection to transform contingent/conformant problems into classical ones. CLG and SDR provided complete approximations for contingent planning domains, at the cost of generating larger classical projections. HCP, on the other hand, generates a simpler projection to capture all the possible paths to the goal so as to measure the usefulness of sensing actions. If the contingent problem is simple, HCP behaves exactly as K-Planner with the additional incorporation of landmark heuristics. In the general case, however, the projection used by HCP is incomplete but can be used in any contingent planning domain, regardless of its contingent width, and the landmarks can be used as a heuristic.

HCP was experimentally compared to CLG, MPSR and K-Planner, using FF as the underlying classical planner. Based on the evaluation, HCP is significantly faster than all other planners in most of the problems; it also solved more instances than the rest of the planners and, in general, generated shorter solutions. It is also important that even in domains that do not involve useful landmarks, HCP worked well due to the rest of its heuristic elements.

Finally, Palacios, *et al.*[10] present an offline contingent planning method based on Brafman and Shani;[79] specifically, two translations of contingent problems into classical ones are presented, being polynomial in the number of possible initial states. One translation converts the sequence of actions in any topological traversal of a policy tree into a classical plan, while the other takes into account only the actions in a single traversal of the policy tree. The latter utilizes a stack element of size k, in which the states that predict an atom p to be false are pushed so as to be dealt with at a later time. This translation is polynomial in k and a stack of size k can be used to generate contingent plans with branches that comprise up to k observations. However, the resulting classical plans may be exponential in k.

Fast Downward, LAMA 2011,[81] FF and the SIW planner[82] were evaluated using both translations; FF performs the worst in general, while LAMA is the fastest and, as such, it was the underlying classical planner of choice in the evaluation of this method against Contingent-FF, MBP, POND, CLG and DNFct. CLG and DNFct clearly outperform the method in Palacios *et al.*,[10] which performed similarly only in relation to earlier contingent planners, such as Contingent-FF, POND and MBP as to the coverage of problems and the generated solutions.

3. WEB SERVICE COMPOSITION METHODS

WSC aims at generating new composite web services that exhibit functionalities not supported by any existing web service. In its simplest form this is achieved by linking existing atomic or composite web services in sequence in order to create new ones. In more complex forms, other control structures can be used, to link services in parallel or using alternative ones, whereas the problem formulation can be extended so as to include further constraints, *e.g.*, QoS (Quality of Service) optimization ones.[83,84] Due to the vast amount of available web services and the burden of searching for the appropriate web service for each goal separately, it is vital that the process of web service composition is automated. As such, AI planning has been utilized to tackle the WSC problem. Moreover, since web services operate in an ever-changing environment, they do not always produce the expected results, they are not always active and they do not have the same interface or even provide the same functionalities. For this reason, the problem of automatic web service composition is more realistically formulated as non-deterministic. In relation to the previous sections, we provide an overview of existing methods dealing with the problem of WSC, focusing on those that incorporate non-determinism. We assume that the reader is familiar with the fundamentals of web services.[85]

A plethora of surveys of WSC methods exist, with different scope, categorization and focus; the majority of them classify methods based on the AI planning techniques used.[86–89] Fewer surveys review other methods such as workflow ones,[90,91] or in relation to manual and semi-automatic ones,[92] whereas, lately, several surveys review WSC methods that support non-determinism, mainly as part of a wider categorization.[93,94] The main focus of this section is automated non-deterministic WSC methods that view the problem as one in which web services are simple planning operators and use planning techniques to tackle it. In the interest of providing a complete review of the current state-of-the-art WSC methods, and since non-deterministic WSC methods are scarce in the literature, other methods such as ones that are semi-automatic or deterministic in their problem formulation are also reviewed. However, methods that are

radically different from the scope of this article, *e.g.*, those that consider web services as stateful[95] or model them as automata,[96] will not be reviewed. Table 4 presents the reviewed methods along with their features.

Table 4. Synopsis of web service composition methods

Method	Determinism	Based on	Heuristic/ Stratified	Evaluation
OWLS-Xplan	Deterministic	FF + HTN	•/-	IPC3
PORSCE II	Deterministic	LPG-td / JPlan	•/-	Single domain
WSPR	Deterministic	Two-step search	•/-	EEE05 -ICEBE05
Oh *et al.*[106]	Deterministic	A*	•/-	-
Huang *et al.*[109]	Deterministic	Two-step forward filtering	-/-	Web Service Challenge 2009
AWSP	Deterministic	A* Backward/forward search	•/-	WSBen
Zúñiga *et al.*[113]	Deterministic	ND-SHOP2	-/-	Case study implementation
Zuñiga *et al.*[115]	Deterministic	SHOP2	-/-	Case study implementation
Wagner *et al.*[116]	Alternative services	Keikaku	-/-	Random problem generator – Against QSynth
Deng *et al.*[118]	Alternative services Multiple solutions	BTSC-DFS	-/•	China Web Service Cup - Web Service Challenge 2009
Wagner *et al.*[121]	Alternative services Multiple solutions	Genetic algorithm	-/-	Custom synthetic problems
Birchmeier[122]	Multiple solutions	A* and beam search	•/-	-
Rodriguez-Mier *et al.*[108]	Multiple solutions	A*Backward search	•/•	Web Service Challenge 2008
Cui *et al.*[126]	Multiple solutions	Viterbi algorithm	-/•	Custom problems - Against human expert
Meyer and Weske[128]	Non-deterministic	EHC	•/-	-
Hoffmann *et al.*[131-133]	Non-deterministic	Conformant-FF	•/-	Two artificial problems - Against the DLVK tool
Mediratta and Srivastava[136]	Non-deterministic	A*	•/-	Two planning problems – Against MBP and SGP
Dacosta *et al.*[138]	Non-deterministic	-	•/•	Implemented prototype
Zou *et al.*[139]	Non-deterministic	FF and SatPlan06	•/-	ICEBE05 - Against WSPR
Wang *et al.*[141]	Non-deterministic	GraphPlan	-/-	ICEBE05

3.1 Deterministic methods

One of the first methods to utilize the connection between AI planning and WSC was OWLS-Xplan;[97] the original - deterministic - WSC domain was translated to a planning one and a custom hybrid planner was used. The planner, Xplan, combines the use of FF with a simple form of HTN decomposition. A replanning module was incorporated to deal with any non-deterministic changes to the world state. During execution, if the current action cannot be applied to the current state, the planner searches for an alternative path from the current state to the goals. The approach was evaluated using the benchmarks of the 2003 IPC3,[98] against the top four - strictly planning - performing participants of that year's competition (FF, SimPlanner, TLPlan,[99] and SHOP2), having the best performance in regard to the average plan quality, and solving nearly all the given problem instances.

PORSCE II[100] is similar to OWLS-XPlan; the main difference being that the planning domain is semantically enhanced. This enhancement is used in case exact solutions to the problem cannot be found by adding semantically equivalent or relevant concepts to the original domain. The system allows the replacement of a single web service with another one or a set of services that generate the same effects, however, this task is not incorporated into the planning process/execution itself. The system was subsequently enhanced by adopting a unifying methodology,[101] allowing for the use of various semantic and non-semantic web service description standards for WSC, without the underlying method

being aware of the specific one used. The method was evaluated in Hatzi *et al.*[100] through synthetic problems based on the SemWebCentral OWL-S Test Collection (`http://semwebcentral.org/frs/?group_id=89`), with the preprocessing, transformation and planning time in relation to the web services in the test problems being used as the evaluation metrics. LPG-td[102] exhibited the best performance among the planners, with the results without semantic relaxation being similar to those with the best semantic relaxation metric.

WSPR (Web Service Planner)[103] is based on a two-step search polynomial time algorithm. First, the cost of achieving the values of individual parameters is computed, assuming that each web service can be invoked with a non-negative cost; the forward search starts from the initial state describing the user's request, and then an optimal plan is approximated through the use of regression search, using the results of the first step as guidance. The backward search is heuristic, having the goal of minimizing the length of the solution, *i.e.*, the number of web services in the plan. Moreover, it is assumed that a web service contributing more to match a sub-goal earlier in the search will also be helpful to reach the initial state faster and, as a result, such services are favored during search. Simultaneously, the heuristic tries to avoid choosing web services that are only partial matches.

WSPR was evaluated using EEE05 (`http://www.comp.hkbu.edu.hk/~eee05/contest/`) and ICEBE05 (`http://www.comp.hkbu.edu.hk/~ctr/`); as the method is targeted for WSDL services[104] and UDDI registries,[105] both test sets only use WSDL files and syntactic - not semantic - matching. The evaluation criteria are the planner's total running time and the number of web services in the plans. The number of web services in a solution, the total number of available web services and the total number of parameters in the problem are the factors that affect the method's performance, which is mainly dependent on the efficiency of the forward search, as it has much worse computation time than the regression one.

Oh *et al.*[106] extended WSPR by using A*;[107] the aggregated QoS value up to the current node was used as its path-cost function, and nodes whose contribution to find the remaining parameters is the maximum were favored. The use of A* allows to generate optimal compositions if the heuristic is admissible; however as Rodríguez-Mier *et al.*[108] note the drawback of the heuristic used in Oh *et al.*[106] is that it is not informative in the case when only the last services of a composition produce all the required parameters.

Huang *et al.* also propose a two-step method based on a forward filtering algorithm to prune the search space, followed by a backward – dynamic – search that computes the optimal QoS values for the WSC problem.[109] The filtering algorithm removes two types of services; first, services that cannot be enabled by the available inputs and, second, services that provide the required outputs but not with the optimal value for the response time or throughput. The method was evaluated using problems from the Web Service Challenge 2009.[110] The average search time of the method over the entire set of test problems was used, without the specifics for each problem being provided. The time to find a solution appears to increase linearly with the amount of services comprising the problem.

AWSP (Automatic Web Service Planner) is a search method that utilizes two implementations of the A* algorithm and a custom heuristic state space search.[111] It can function either as a forward state search planner or as a backwards one, using two heuristics that rely on the concept of parameter distance; parameter distance between two parameters is defined to be equal to one if they are part of the input parameters and the output parameters of a single web service, respectively, and infinite if no invocation between them is possible.

If longer paths exist that require the generation of intermediate parameters, it is equal to the length of the shortest path of invocation between them. This method requires precomputing the parameter distance between any two pairs, thus if any modification is made to the web service registry, *e.g.*, a web service is added or removed, this computation has to be done again.

The method is evaluated with the WSBen benchmark,[112] using as evaluation metrics the running time to find the solution, the number of web services in it and the states that were expanded during search. The backwards search proved substantially better than the forward one, as the state space is much larger in the latter. Additionally, the authors favor their own algorithm instead of A*, as it generates solutions faster and with a quality similar to A*, although not optimal. Rodríguez-Mier *et al.*[108] argue that since Wu *et al.*[111] do not utilize stratified methods, the search space size cannot be quickly reduced. Wu *et al.*,[111] on the other hand, argue that although stratified methods are efficient, they can only tackle WSC problems in which the web services are only information-providing and not world altering.

Zúñiga *et al.*[113] make use of AI planning in conjunction with WSMO (Web Services Modeling Ontology).[114] A translation from the web service domain to the planning one occurs before planning and then the resulting plans are evaluated to choose the best one, considering the services' non-functional properties. The planner used is ND-SHOP2, which generates all the policies that solve the original prob-

lem; apart from the best plan, though, the rest are not taken into consideration, and if a failure occurs during execution, replanning generates a new plan from scratch. A prototype was implemented, without, however, any quantitative evaluation of the method. Zuñiga *et al.*[115] build upon this method; they provide a loosely coupled architecture and utilize SHOP2 as the underlying planner, providing an evaluation of the method based on a single case study, without quantitative results.

3.2 Middle ground methods

A middle ground between straightforward deterministic methods and those that support non-determinism groups similar web services in their solutions and uses them as alternatives. Another one is methods that generate multiple - or all possible - solutions to a problem, rank them according to some QoS criteria, and output a single optimal solution.

Wagner *et al.*,[116] in contrast to Zuñiga *et al.*,[115] argue that the basic problem of most WSC methods is that they only tackle the problem of flexibility, *i.e.*, automatic WSC, while also either trying to maximize the reliability of the produced plans, or achieving QoS goals, but not both. Their method generates workflows incorporating alternative "backup" web services. It first generates clusters containing similar web services, thereby pruning the search space. A regression planner based on iterative deepening depth-first search is then used in order to generate the workflows, taking the QoS of each cluster into account. This method is compared favorably against an extension of QSynth[117] based on a random problem generator; however, in case there are few variations of the services in the registry, the method is less efficient, generating longer workflows with low reliability.

Deng *et al.* present a QoS-based WSC method generating the top-k solutions of WSC problems.[118] The original problem is split into mutually independent smaller ones that can be solved concurrently. Then the best k solutions are returned in relation to some QoS criterion, with the rules in Zeng *et al.*[119] being used to compute the aggregate QoS for web services either in a sequence execution path or in multiple parallel paths. The solution is returned as a DAG with the sole criterion being its response time. The first step is to transform the original set of web services into a rule repository, with each web service comprising a rule that can generate both the concepts in it and their ancestors. An inverted index of this repository is created, with the rules being indexed by the concepts that can be outputted from them. Web services are identified as either useful or irrelevant, by using an extension of the method in Hennig and Balke.[120] The initial - user provided - concepts create a set and the outputs of the web services that can be executed by the concepts in this set are added to it. This procedure repeats until no more concepts can be added. If the concepts in the set become a superset of the ones in the goal request, then the problem is solvable and the web services that contributed outputs to the set are identified as useful. Otherwise, the goal cannot be met.

Afterwards, the mutually independent tasks are distributed and solution subgraphs for their particular problem are generated using a backtracking algorithm for WSC based on depth-first search (BTSC-DFS). BTSC-DFS returns, for each concept in the request, the set of web services that output it, using the inverted index and a subset of them is chosen to backtrack for. In case two or more web services share the same inputs, they are considered as one common web service. For this reason, the output solution may contain sets of web services instead of single ones.

Wagner *et al.*[121] propose a multi-objective optimization method that outputs multiple alternative solutions to WSC problems through the use of Hierarchical Workflow Graphs (HWG). The problem is solved through a genetic algorithm, with the QoS elements considered being the cost of web services, the required time to output a response, and their reliability.

The method partially orders the current selections based on the notion of strict domination over the objective functions. The problem is modeled as a HWG that contains complex service nodes that represent alternative workflow schemes, each of which is a DAG. The web services are grouped into sets of functionally equivalent ones distinguishable by their QoS properties. Then, a genetic algorithm computes a Pareto-optimal set, among which one solution is chosen based on the input QoS preferences. The method was evaluated using synthetic problems created by Wagner *et al.*,[121] for which optimal solutions could not be computed efficiently even for small problems. Instead, an approximation of optimal solutions is generated.

Birchmeier[122] presents a method that generates k alternative plans, which are then annotated with quality information, in a semi-automatic way. Incomplete plans are computed, using a combination of A^* and beam search, that is, by keeping a limited list of partial plans, a set of k best nodes being considered at each time. This set is ordered according to a parameter representing the degree of similarity between the path of the first selected node and the path of the node that will be valued. The implementation was not subjected to any quantitative evaluation in regard to its efficiency.

Rodriguez-Mier *et al.*[123] used A* to search through a service dependency graph and generate an optimal composition

in terms of the number of web services comprising it. In Rodríguez-Mier *et al.*[108] an optimal and complete backward search method was used to generate multiple optimal solutions to WSC problems, without taking into consideration their non-functional properties. A layered service dependency graph, representing a suboptimal solution, is created dynamically using semantic matching, each layer of which contains all the web services than can be executed with the outputs of the web services in the previous one. A backward heuristic search based on A* is used to generate all optimal solutions that have a different number of services and runpath; the heuristic is simply the layer in which the specific node is placed, *i.e.*, its distance to the initial state. Finally, the solutions found are optimized so that parallelism is maximized and the number of services in each one is minimized. The paths explored are reduced by removing irrelevant services and by replacing, offline, web services that produce the same set of outputs by a representative one. Moreover, during search, the algorithm dynamically detects and combines nodes that can generate equivalent neighbors.

In Rodríguez-Mier *et al.*[108] the WSC algorithm's performance was evaluated based on problems from the Web Service Challenge 2008;[124] optimal solutions were generated in all cases, with the optimizations improving the method's efficiency substantially. The method does not support alternative flows and requires a pre-computed table mapping inputs to the web services that require them. For this reason, each time the description and/or functionalities of a service are modified, the dependency graph is recomputed, from the service's layer up to the last one. In Rodríguez-Mier *et al.*[125] the method was paired with an existing service registry utilizing Linked Data principles and semantic reasoning along with various service discovery and matchmaking configurations. It was shown that unless several indexing optimizations are incorporated, the system performs poorly; furthermore, the time required for the discovery and matchmaking phases dominates that of WSC.

Finally, Cui *et al.*[126] present a deterministic method that generates all feasible solutions based on a service graph that represents the web service workflows and outputs a single optimal solution based on dynamic programming. The Viterbi algorithm[127] is used to select among the possible plans, with multiple QoS attributes, such as the web services' price, response time and reliability being considered. The method is evaluated with no information being provided as to the test set's problems or their size. Its performance is evaluated against an optimal solution selected by a human expert and the average value of the feasible solutions computed earlier, taking into account the number of web services comprising the solutions.

Generating all possible solutions only to discard all but one in the end is highly inefficient, as, having alternative plans at hand, one can generate a contingent plan that works despite non-determinism in the domain. The next section presents various methods that utilize such a methodology, along with other non-deterministic WSC methods.

3.3 Non-deterministic and contingent planning methods

Meyer and Weske[128] proposed one of the first methods that took uncertainty into account during WSC, assuming that it was present in the problem's initial state and the web services' effects. For this reason, alternative control flow elements were incorporated in the composition process. The method's planning module is based on forward heuristic search in state space, particularly enforced hill-climbing (EHC).[22] As EHC does not support non-determinism, its state transition function and the way states are handled were modified. The length of the solution to a simplified version of the original problem using Graphplan was used as an admissible heuristic;[129] however, the planning method was significantly slower than either Conformant-FF[130] or Contingent-FF,[47] partly due to the larger search space of the possible parallel invocations and the method's non-optimized representation of states.

In Hoffmann *et al.*,[131] as well as in subsequent work,[132, 133] web service application is considered as a belief update operation and concept subsumption relations are modeled through forward effects, in order to allow for the use of ontologies into AI planners.[134] Forward effects in WSC are assumed to be present when all the ramifications of a web service's application concern only propositions involving at least one new constant. When problems fall into the two special cases of WSC under uncertainty that are identified in Hoffmann *et al.*[133] they are tractable and allow for a compilation of the original WSC problem into conformant planning. Conformant-FF was modified to consider on-the-fly output constants and used to evaluate the method with and without EHC and the helpful actions pruning optimizations[133] against the DLVK tool.[135] The total runtime, number of search states expanded and the length of the plans were the evaluation metrics. DLVK was significantly slower than CFF and solved considerably less problem instances.

A semi-automatic, interactive and limited, contingent WSC method is presented in Mediratta and Srivastava[136] that allows the intervention of expert users in order to control the search over the possible plans. Users are allowed to insert default branches instead of those that they are not interested in, as well as define soft constraints. In essence, in order to allow incomplete modeling, user acceptable plans are allowed that specify a subset of the branches that are guaranteed to lead to

the goal.[133] The search space consists of belief states in an AND-OR graph; sensing actions form the AND part of the graph and the rest of the actions the OR part of it. Users can specify which part of the search space is considered good with the help of a cost measure that estimates heuristic distance, so that the search is focused on good states. In this way, the AND part of the graph can be pruned efficiently, and the Planning Graph heuristic[137] can be used to prune the rest of the graph. Search is conducted in a forward fashion, while the heuristic is computed backwards, only taking into account relevant actions. The whole planning process is complete provided that the user does not define external specifications that prune the search space.

The method was compared to a naïve planning method that generated a complete plan under all conditions, filtering specific branches afterwards based on the user's constraints, as well as against MBP and SGP, based on planning domains from the benchmark set in SGP. The method performed significantly better than the naïve implementation, comparably with MBP, and consistently considerably worse than SGP.

Dacosta *et al.*[138] present a method that produces robust plans through the generation of contingency plans, allowing for web services that achieve the same semantic tasks. The notions of useful and redundant operations are utilized, using a different method to distinguish between them than in Deng *et al.*;[118] then, only useful web services' operations can be selected, thus effectively restricting the search space. The method outputs a graph comprising control flow constructs and calls to web services' operations, containing all the possible contingency plans, utilizing a stratified method. Initially, the initial state and the goals are given and, then, several rules are applied continuously, until no rule can be applied anymore, or a plan is feasible. This step restricts the number of possible execution paths that the planner has to search among, and outputs the useful operations' set if a plan is possible. Then, the operations that generate each output, effect and activity populate one vector each, and the sets of vectors are used to output all the possible paths to the goal; that is, all paths that produce every output of the original request are generated, comprising one element of each vector.

Next, redundant operations are removed from the paths, the paths are reordered in relation to their preconditions and input dependencies, and the set of paths is ordered from the best – the one containing the smaller number of services – to the worst. The resulting graph, containing all the possible execution paths, can only fail if an operation fails and there is no other brother or uncle of the node that can be executed. This process is similar to the one in Rodríguez-Mier *et al.*[108] Dacosta *et al.* describe a prototype as well as an example of the method on a test problem,[138] without providing evaluation results.

Zou *et al.*[139] proposed a method that determinizes the original problem and then solves it using either FF or SatPlan06,[140] similarly to the methods presented in Section 2.3. The method does not take into account semantic content and generates plans for multiple participants that collaborate with each other to achieve a common goal. Users input explicitly defined contingencies; then the original WSC problem is transformed into a planning one and deterministically solved. A dependency graph is built based on this solution,[139] which is used to create a master plan that is projected to a distributed one. The method is compared to WSPR based on problems from the ICEBE05 WSC challenge; WSPR is shown to be slower, despite its much simpler problem formulation and goals, mainly due to the efficiency of FF and SatPlan06 used in Zou *et al.*[139]

Finally, Wang *et al.*[141] presented a WSC method that took uncertainty in the actions' effects into account, through a modification of Graphplan. The method allows for operators with multiple groups of effects as well as branches in the output solutions. The method was evaluated using the ICEBE05 test set; in the experiments that did not include uncertainty, the algorithm's performance depended on the number of the web services comprising the problem, their number of parameters and the problem's solution length. In the problems that incorporated uncertainty, after modifying the ICEBE05 problems by adding preconditions and multiple effects, the evaluation demonstrated that the problems with uncertainty usually require approximately double the time than the respective deterministic ones. The method's performance relies on the number and location of the non-deterministic actions in the outputs solution and their number of uncertain effects.

4. Conclusion

This article comprises a thorough and up-to-date review of non-deterministic planning methods with the goal to be integrated in automated WSC systems.

Although the need for the incorporation of non-determinism in the WSC process has been acknowledged in the literature, there are relatively few methods that attempt to tackle it. Despite the advances in the non-deterministic planning research field, presented in this article, the majority of WSC methods still focus in using classical planners in conjunction with a deterministic problem formulation. Recently, methods that incorporate the generation of multiple solutions or alternative web services in them have been proposed, as well as

a few that adopt non-deterministic formulations. However, most of the methods in the literature, either utilize custom search algorithms or incorporate classical planners in the WSC process. Methods in the former category are usually based on A*,[106, 108, 122, 136] while examples of the latter are OWLS-Xplan, PORSCE II and the methods in Zuñiga *et al.*[115] and Wang *et al.*[141]

Only a few methods actually exploit the results from the non-deterministic planning research, by incorporating the determinization schemes proposed by FF-Replan, such as in Zou *et al.*,[139] or through direct use of non-deterministic planners.[113, 131] None, though, actually use any of the state-of-the-art contingent planning methods that were highlighted in our review. This fact complicates their implementation and restricts their scalability; moreover, along with the relative immaturity of the WSC research and the diversity of web service standards and methods used, it also hinders the direct comparison between WSC methods.

It is our view that non-determinism is inherent in WSC and that in the spirit of web services themselves, WSC methods should also have a modular architecture and efficiently re-use existing tools. This article has drawn attention to the progress non-deterministic methods have made in terms of scalability, speed, and generality of the adopted formalizations. Moreover, we have highlighted the current top non-deterministic methods exploited by WSC systems; we expect that in the near future, such tools will be integrated directly into WSC approaches so as to provide more efficient applications.

ACKNOWLEDGEMENTS

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

REFERENCES

[1] Planning: Three Models, one Algorithm. Proceedings of the AIPS'98 Workshop on Planning as Combinatorial Search; 1998.

[2] Fu J, Ng V, Bastani FB, *et al*. Simple and fast strong cyclic planning for fully-observable non-deterministic planning problems. Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI); 2011.

[3] Kuter U. Pushing the limits of AI planning. Proceedings of the Doctoral Consortium of the 14th International Conference on Automated Planning and Scheduling (ICAPS); 2004.

[4] Bryce D, Buffet O. International Planning Competition Uncertainty part: benchmarks and results. Proceedings of the 6th International Planning Competition (IPC); 2008.

[5] Ramírez M, Yadav N, Sardiña S. Behavior composition as fully observable non-deterministic planning. Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS); 2013.

[6] Cimatti A, Pistore, M, Roveri M, *et al*. Weak, strong, and strong cyclic planning via symbolic model checking. Artificial Intelligence. 2003; 147(1-2): 35-84.

[7] Muise C, Belle V, McIlraith SA. Computing contingent plans via fully observable non-deterministic planning. Proceedings of the 24th ICAPS Workshop on Models and Paradigms for Planning under Uncertainty; 2014.

[8] Bertoli P, Cimatti A, Roveri M, *et al*. Strong Planning under Partial Observability. Artificial Intelligence. 2006; 170(4-5): 337-84. http://dx.doi.org/10.1016/j.artint.2006.01.004

[9] Bonet B, Geffner H. Planning under partial observability by classical replanning: Theory and experiments. Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI); 2011.

[10] Palacios H, Albore A, Geffner H. Compiling contingent planning into classical planning: New translations and results. Proceedings of the 24th ICAPS Workshop on Models and Paradigms for Planning under Uncertainty; 2014.

[11] Geffner H, Bonet B. A concise introduction to models and methods for automated planning, 1st ed. Morgan, Claypool Publishers; 2013.

[12] Onder N, Pollack ME. Conditional, probabilistic planning: a unifying algorithm and effective search control mechanisms. Proceedings of the 16th National Conference On Artificial Intelligence (AAAI); 1999.

[13] Pryor L, Collins G. Planning for Contingencies: A decision-based approach. Journal of Artificial Intelligence Research (JAIR). 1996; 4: 287-339.

[14] Majercik SM. APROPOS2: Approximate probabilistic planning out of stochastic satisfiability. Proceedings of the AAAI Workshop on Probabilistic Approaches in Search of the 18th National Conference on Artificial Intelligence; 2002.

[15] Meuleau N, Smith DE. Optimal limited contingency planning. Proceedings of the 19th conference on Uncertainty in Artificial Intelligence; 2003.

[16] Bonet B. Conformant plans and beyond: Principles and complexity. Artificial Intelligence. 2010; 174(3-4): 245-69.

[17] Bonet B. Bounded branching and modalities in non-deterministic planning. Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS); 2006.

[18] Kaelbling LP, Littman ML, Cassandra AR. Planning and acting in partially observable stochastic domains. Artificial Intelligence. 1998; 101: 99-134. http://dx.doi.org/10.1016/S0004-3702(98)00023-X

[19] Hyafil N, Bacchus F. Conformant probabilistic planning via CSPs. Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS); 2003.

[20] Yoon S, Fern A, Givan R. FF-Replan: A baseline for probabilistic planning. Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS); 2007.

[21] Younes H, Littman M. Proceedings of the International Probabilistic Planning Competition [Internet]; 2007 [cited 2015 Aug 1]. Available from: http://www.cs.rutgers.edu/~mlittman/topics/ipc04-pt/

[22] Hoffmann J. FF: The Fast-Forward planning system. AI Magazine. 2001; 22(3): 57-62.

[23] Hoffmann J, Nebel B. The FF planning system: fast plan generation through heuristic search. Journal of Artificial Intelligence Research (JAIR). 2001; 14: 253-302.

[24] Kuter U, Nau DS, Reisner E, *et al*. Using classical planners to solve nondeterministic planning problems. Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS); 2008.

[25] Teichteil-Koenigsbuch F, Infantes G, Kuter U. RFF: A robust, FF-based MDP planning algorithm for generating policies with low probability of failure. Proceedings of the 3rd International Planning Competition (IPPC-ICAPS); 2008.

[26] Yoon S, Fern A, Givan R, *et al*. Probabilistic planning *via* determinization in hindsight. Proceedings of the 23rd Conference on Artificial Intelligence (AAAI); 2008.

[27] Yoon S, Ruml W, Benton J, *et al*. Improving determinization in hindsight for online probabilistic planning. Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS); 2010.

[28] Buffet O, Aberdeen D. The Factored Policy Gradient Planner. Proceedings of the Probabilistic Planning Track of the 2006 International Planning Competition; 2006.

[29] Jiménez S, Coles A, Smith A. Planning in probabilistic domains using a deterministic numeric planner. Proceedings of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSig); 2006.

[30] Foss J, Onder N, Smith D. Preventing unrecoverable failures through precautionary planning. Proceedings of the 17th ICAPS Workshop on Moving Planning and Scheduling Systems into the Real World; 2007.

[31] Dearden R, Meuleau N, Ramakrishnan S, *et al*. Incremental contingency planning. Proceedings of the 13th ICAPS Workshop on Planning under Uncertainty; 2003.

[32] Bonet B, Geffner H. Planning with incomplete information as heuristic search in belief space. Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling (ICAPS); 2000.

[33] Korf R. Real-time heuristic search. Artificial Intelligence. 1990; 42: 189-211. http://dx.doi.org/10.1016/0004-3702(90)90054-4

[34] Barto A, Bradtke S, Sing S. Learning to act using real time dynamic programming. Artificial Intelligence. 1995; 72: 81-138. http://dx.doi.org/10.1016/0004-3702(94)00011-O

[35] Bonet B, Geffner H. Planning as heuristic search. Artificial Intelligence. 2001; 129(1-2): 5-33. http://dx.doi.org/10.1016/S0004-3702(01)00108-4

[36] Bonet B, Geffner H. mGPT: A Probabilistic Planner Based on Heuristic Search. Journal of Artificial Intelligence Research (JAIR). 2005; 24: 933-44.

[37] Bertoli P, Cimatti A, Pistore M, *et al*. MBP: a Model Based Planner. Proceedings of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information; 2001.

[38] Bryant RE. Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys. 1992; 24(3): 293-318. http://dx.doi.org/10.1145/136035.136043

[39] To ST, Son TC, Pontelli E. Contingent Planning as AND/OR forward search with disjunctive representation. Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS); 2011.

[40] Jensen R, Veloso M. OBDD-based universal planning for synchronized agents in non-deterministic domains. Journal of Artificial Intelligence Research (JAIR). 2000; 13: 189-226.

[41] Kabanza F, Barbeau M, St-Denis R. Planning control rules for reactive agents. Artificial Intelligence. 1997; 95(1): 67–113. http://dx.doi.org/10.1016/S0004-3702(97)00031-3

[42] Rintanen J. Constructing conditional plans by a theorem-prover. Journal of Artificial Intelligence Research (JAIR). 1999; 10: 323-52.

[43] Weld DS, Anderson CR, Smith DE. Extending Graphplan to handle uncertainty and sensing actions. Proceedings of the 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI); 1998.

[44] Rintanen J. Backward plan construction under partial observability. Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems; 2002.

[45] Rintanen J. Product representation of belief spaces in planning under partial observability. Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI); 2003.

[46] Bryce D, Kambhampati S, Smith DE. Planning in belief space with a labelled uncertainty graph. Proceedings of the 19th AAAI Workshop on Learning and Planning in Markov Decision Processes; 2004.

[47] Hoffmann J, Brafman R. Contingent planning via heuristic forward search with implicit belief states. Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS); 2005.

[48] Nilsson N. Principles of Artificial Intelligence. Tioga; 1980.

[49] Bryce D, Kambhampati S, Smith DE. Planning graph heuristics for belief space search. Journal of Artificial Intelligence Research (JAIR). 2006; 26: 35-99.

[50] Bouguerra A, Karlsson L. PC-SHOP: A probabilistic-conditional hierarchical task planner. Intelligenza Artificiale. 2005; 2(4): 44-50.

[51] Nau DS, Cao Y, Lotem A, *et al*. SHOP: Simple hierarchical ordered planner. Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI); 1999.

[52] Kuter U, Nau DS. Forward-chaining planning in nondeterministic domains. Proceedings of the 19th National Conference on Artificial Intelligence (AAAI); 2004.

[53] Nau D, Au T, Ilghami O, *et al*. SHOP2: An HTN planning system. Journal of Artificial Intelligence Research (JAIR). 2003; 20: 379-404.

[54] Kuter U, Nau DS, Pistore M, *et al*. A hierarchical task-network planner based on symbolic model checking. Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS); 2005.

[55] Alford R, Kuter U, Nau DS, *et al*. Plan aggregation for strong-cyclic planning in nondeterministic domains. Artificial Intelligence. 2014; 216: 206-32. http://dx.doi.org/10.1016/j.artint

[56] Kissmann P, Edelkamp S. GAMER: Fully-Observable Non-Deterministic Planning via PDDL-Translation into a Game. Proceedings of the FOND track of the 6th International Planning Competition (IPPC) of the 18th ICAPS; 2008.

[57] Kissmann P, Edelkamp S. Solving fully-observable non-deterministic planning problems via translation into a general game. Proceedings of the 32nd Annual German Conference on AI (KI); 2009. http://dx.doi.org/10.1007/978-3-642-04617-9_1

[58] Fu J, Jaramillo CA, Ng V, *et al*. Fast strong planning for FOND problems with multi-root directed acyclic graphs. Proceedings of

the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI); 2013. http://dx.doi.org/10.1109/ICTAI.2013.23

[59] Jaramillo CA, Fu J, Ng V, *et al*. Fast Strong Planning for FOND Problems with Multi-Root Directed Acyclic Graphs. International Journal of Artificial Intelligence Tools. 2014; 23(6). http://dx.doi.org/10.1142/S021821301

[60] Muise C, McIlraith SA, Beck JC. Improved non-deterministic planning by exploiting state relevance. Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS); 2012.

[61] Helmert M. The Fast Downward planning system. Journal of Artificial Intelligence Research (JAIR). 2006; 26(1): 191-246.

[62] Little I, Thiebaux S. Probabilistic planning vs replanning. Proceedings of the 17th ICAPS Workshop on the International Planning Competition: Past, Present and Future; 2007.

[63] Camacho A, Muise C, Ganeshen A, *et al*. From FOND to probabilistic planning: Guiding search for quality policies. Proceedings of the 25th ICAPS Workshop on Heuristic Search and Domain Independent Planning (HSDIP'15); 2015.

[64] Winterer D, Mattmüller R, Wehrle M. Stubborn sets for fully observable nondeterministic planning. Proceedings of the 25th ICAPS Workshop on Model Checking and Automated Planning (MOCHAP'15); 2015.

[65] Wehrle M, Helmert M, Shleyfman A, *et al*. Integrating Partial Order Reduction and Symmetry Elimination for Cost-Optimal Classical Planning. Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI); 2015.

[66] Hansen EA, Zilberstein S. LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence. 2001; 129(1-2): 35-62. http://dx.doi.org/10.1016/S0004-3702(01)00106-0

[67] Buffet O, Aberdeen D. FF+FPG: Guiding a policy-gradient planner. Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS); 2007.

[68] To ST, Pontelli E, Son TC. A conformant planner with explicit disjunctive representation of belief states. Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS); 2009.

[69] Albore A, Palacios H, Geffner H. A translation-based approach to contingent planning. Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI); 2009.

[70] Maliah S, Brafman R, Karpas E, *et al*. Partially observable online contingent planning using landmark heuristics. Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS); 2014.

[71] Russell S, Norvig P. Artificial intelligence: A modern approach, 2nd ed, Prentice-Hall; 2002.

[72] Albore A, Geffner H. Acting in partially observable environments when achievement of the goal cannot be guaranteed. Proceedings of the 19th ICAPS Workshop on Planning and Plan Execution for Real-World Systems; 2009.

[73] Keyder E, Geffner H. Heuristics for planning with action costs revisited. Proceedings of the 18th European Conference on AI (ECAI); 2008.

[74] Shani G, Brafman RI. Replanning in domains with partial information and sensing actions. Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI); 2011.

[75] Shani G, Brafman R, Maliah S, *et al*. Heuristics for planning under partial observability with sensing actions. Proceedings of the 23rd ICAPS Workshop on Heuristics and Search for Domain-independent Planning (HSDIP'13); 2013.

[76] Bonet B, Geffner H. Flexible and Scalable Partially Observable Planning with Linear Translations. Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI); 2014.

[77] Brafman RI, Shani G. Replanning in domains with partial information and sensing actions. Journal of Artificial Intelligence Research (JAIR). 2012; 45: 565-600.

[78] Palacios H, Geffner H. Compiling uncertainty away in conformant planning problems with bounded width. Journal of Artificial Intelligence Research (JAIR). 2009; 35: 623-75.

[79] Brafman RI, Shani G. A multi-path compilation approach to contingent planning. Proceedings of the 26th AAAI Conference on Artificial Intelligence; 2012.

[80] Hoffmann J, Porteous J, Sebastia L. Ordered landmarks in planning. Journal of Artificial Intelligence Research (JAIR). 2004; 22(1): 215-78.

[81] Richter S, Westphal M. The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research (JAIR). 2010; 39(1); 127-77.

[82] Lipovetzky N, Geffner H. Width and serialization of classical planning problems. Proceedings of the 20th European Conference of Artificial Intelligence (ECAI); 2012.

[83] Rao J, Su X. A survey of automated web service composition methods. Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC); 2003.

[84] George Markou, Ioannis Refanidis. Composing Semantic Web Services Online and an Evaluation Framework. International Journal on Advances in Internet Technology. 2013; 6(3-4): 114-31.

[85] Heymans S, Hoffmann J, Marconi A, *et al*. Semantic web services fundamentals, Chapter in Handbook of Service Description: USDL and Its Methods. Springer-Verlag; 2011.

[86] Chan M, Bishop J, Baresi L. Survey and comparison of planning techniques for web services composition - Technical report. University of Pretoria (South Africa); 2007.

[87] Peer J. Web service composition as AI planning: A survey - Technical report. University of St. Gallen (Switzerland); 2005.

[88] Oh SC, Lee D, Kumara SRT. A comparative illustration of AI planning-based web services composition. Proceedings of the ACM SIGecom Exchanges; 2005.

[89] Digiampietri LA, Pérez-Alcázar JJ, Medeiros CB. AI planning in web services composition: A review of current approaches and a new solution. Proceedings of the 6th Encontro Nacional de Inteligencia Artificial (ENIA); 2007.

[90] Eid MA, Alamri A, El-Saddik A. A reference model for dynamic web service composition systems. International Journal of Web and Grid Services. 2008; 4(2): 149-68. http://dx.doi.org/10.1504/IJWGS

[91] Tabatabaei SGH, Kadir WMNW, Suhaimi I. An evaluation of current approaches for web service composition. Proceedings of the 3rd International Symposium on Information Technology (ITSim); 2008.

[92] Sheng QZ, Qiao X, Vasilakos AV, *et al*. Web services composition: A decade's overview, Information Sciences. 2014; 280: 218-38. http://dx.doi.org/10.1016/j.ins.2014.04.054

[93] Bartalos P, Bieliková M. Automatic dynamic web service composition: a survey and problem formalization. Computing and Informatics. 2011; 30(4): 793-827.

[94] D'Mello DA, Ananthanarayana VS, Salian S. A review of dynamic web service composition techniques. Proceedings of the 1st International Conference on Computer Science and Information Technology (CCSIT 2011); 2011.

[95] Bertoli P, Pistore M, Traverso P. Automated composition of Web services via planning in asynchronous domains. Artificial Intelligence. 2010; 174(3-4): 316-61. http://dx.doi.org/10.1016/j.artint.2009.12.002

[96] Giacomo GD, Mecella M, Patrizi F. Automated service composition based on behaviors: the roman model, Chapter in Web Services Foundations. Springer; 2014.

[97] Klusch M, Gerber A. Evaluation of service composition planning with OWLS-XPlan. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IATW '06); 2006.

[98] Long D, Fox M. The 3rd International Planning Competition: Results and Analysis. Journal of Artificial Intelligence Research (JAIR). 2003; 20: 1-59.

[99] Bacchus F, Kabanza F. Using temporal logics to express search control knowledge for planning. Artificial Intelligence. 2000; 116(1-2): 123-91.

[100] Hatzi O, Vrakas D, Nikolaidou M, *et al*. An integrated approach to automated semantic web service composition through planning. IEEE Transactions on Services Computing. 2012; 5(3): 319-32.

[101] Hatzi O, Nikolaidou M, Vrakas D, *et al*. Semantically aware web service composition through AI planning. International Journal on Artificial Intelligence Tools. 2015; 24(1). http://dx.doi.org/10.1142/S021821301

[102] Gerevini A, Saetti A, Serina I. An approach to temporal planning and scheduling in domains with predictable exogenous events. Journal of Artificial Intelligence Research (JAIR). 2006; 25: 187-231.

[103] Oh SC, Lee D, Kumara SRT. Web service Planner (WsPr): an effective and scalable web service composition algorithm. International Journal of Web Services Research. 2007; 4(1): 1-23. http://dx.doi.org/10.4018/jwsr.2007010101

[104] Cerami E. Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. First edition: O'Reilly; 2002. Chapter 6, WSDL Essentials; p. 103-33.

[105] Newcomer E. Understanding Web Services: XML, WSDL, SOAP, and UDDI. First edition: Addison-Wesley; 2002. Chapter 5, Finding Web Services: UDDI Registry; p. 110-34.

[106] Oh S, Lee JY, Cheong SH, *et al*. WSPR*: Web-service planner augmented with A* algorithm. Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing (CEC); 2009. http://dx.doi.org/10.1109/CEC.2009.13

[107] Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics. 1968; 4(2): 100-7. http://dx.doi.org/10.1109/TSSC.1968.300136

[108] Rodríguez-Mier P, Mucientes M, Vidal JC, *et al*. An optimal and complete algorithm for automatic web service composition. International Journal of Web Services Research. 2012; 9(2): 1-20. http://dx.doi.org/10.4018/jwsr.2012040101

[109] Huang Z, Jiang W, Hu S, *et al*. Effective pruning algorithm for QoS-aware service composition. Proceedings of the 11thIEEE Conference on Commerce and Enterprise Computing (CEC); 2009. http://dx.doi.org/10.1109/CEC.2009.41

[110] Kona S, Bansal A, Blake MB, *et al*. WSC-2009: a quality of service-oriented web services challenge. Proceedings of the 11th IEEE Conference on Commerce and Enterprise Computing (CEC); 2009. http://dx.doi.org/10.1109/CEC.2009.80

[111] Wu B, Li Y, Wu J, *et al*. AWSP: an automatic web service planner based on heuristic state space search. Proceedings of the 9th IEEE International Conference on Web Services (ICWS); 2011. http://dx.doi.org/10.1109/ICWS.2011.20

[112] Oh SC, Kil H, Lee D, *et al*. WSBen: a web services discovery and composition benchmark. Proceedings of the 4th IEEE International Conference on Web Services (ICWS); 2006.

[113] Zúñiga JC, Pérez Alcázar JJ, Digiampietri LA. Implementation issues for automatic composition of web services. Proceedings of the 2010 Workshops on Database and Expert Systems Applications (DEXA); 2010.

[114] Fensel D, Lausen H, Polleres A, *et al*. Enabling Semantic Web Services: The Web Service Modeling Ontology. First edition. Berlin, Germany: Springer-Verlag; 2007.

[115] Zuñiga JC, Pérez-Alcázar JJ, Digiampietri LA. A loosely coupled architecture for automatic composition of web services applications. International Journal of Metadata, Semantics and Ontologies. 2014; 9(3): 241-51.

[116] Wagner F, Ishikawa F, Honiden S. QoS-aware automatic service composition by applying functional clustering. Proceedings of the 9th IEEE International Conference on Web Services (ICWS); 2011.

[117] Jiang W, Zhang C, Huang Z, *et al*. QSynth: A tool for QoS-aware automatic service composition. Proceedings of the 8th International Conference on Web Services (ICWS); 2010.

[118] Deng S, Huang L, Tan W, *et al*. Top-K automatic service composition: A parallel method for large-scale service sets. IEEE Transactions on Automation Science and Engineering. 2014; 11(3): 891-905. http://dx.doi.org/10.1109/TASE.2014.2306931

[119] Zeng L, Benatallah B, Ngu AHH, *et al*. QoS-aware middleware for Web services composition. IEEE Transactions on Software Engineering. 2004; 30(5): 311-27.

[120] Hennig P, Balke WT. Highly scalable web service composition using binary tree-based parallelization. Proceedings of the 8th IEEE International Conference on Web Services (ICWS); 2010.

[121] Wagner F, Klein A, Klopper B, *et al*. Multi-objective service composition with time-and input-dependent QoS. Proceedings of the 19th IEEE International Conference on Web Services (ICWS); 2012. http://dx.doi.org/10.1109/ICWS.2012.40

[122] Birchmeier P. Semi-automated semantic web service composition planner supporting alternative plans synthesis and imprecise planning [diploma thesis]. University of Zurich; 2007.

[123] Rodriguez-Mier P, Mucientes M, Lama M. Automatic web service composition with a heuristic-based search algorithm. Proceedings of the 9th IEEE International Conference on Web Services (ICWS); 2011. http://dx.doi.org/10.1109/ICWS.2011.89

[124] Bansal A, Blake MB, Kona S, *et al*. WSC-08: Continuing the web services challenge. Proceedings of the 10th IEEE Conference on E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services; 2008.

[125] Rodriguez-Mier P, Pedrinaci C, Lama M, *et al*. An Integrated Semantic Web Service Discovery and Composition Framework. IEEE Transactions on Services Computing. Forthcoming 2015. http://dx.doi.org/10.1109/TSC.2015.2402679

[126] Cui LZ, Li J, Zheng Y. A dynamic web service composition method based on Viterbi algorithm. Proceedings of the IEEE International Conference on Web Services (ICWS); 2012.

[127] Viterbi AJ. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory. 1967; 13(2): 260-9.

[128] Meyer H, Weske M. Automated service composition using heuristic search. Proceedings of the 4thInternational Conference on Business Process Management (BPM); 2006. http://dx.doi.org/10.1007/11841760_7

[129] Blum A, Furst M. Fast planning through planning graph analysis. Artificial Intelligence. 1997; 90: 281-300. http://dx.doi.org/10.1016/S0004-3702(96)00047-1

[130] Hoffmann J, Brafman R. Conformant planning via heuristic forward search: A new approach. Artificial Intelligence. 2006; 170(6-7): 507-41. http://dx.doi.org/10.1016/j.artint.2006.01.003

[131] Hoffmann J, Bertoli P, Pistore M. Web service composition as planning, revisited: in between background theories and initial state uncertainty. Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI); 2007.

[132] Sirbu A, Hoffmann J. Towards scalable web service composition with partial matches. Proceedings of the 6th IEEE International Conference on Web Services (ICWS); 2008.

[133] Hoffmann J, Bertoli P, Helmert M, *et al*. Message-based web service composition, integrity constraints, and planning under uncertainty: a new connection. Journal of Artificial Intelligence Research (JAIR). 2009; 35: 49-117.

[134] Kaldeli E. Domain-independent planning for services in uncertain and dynamic environments [dissertation]. University of Groningen; 2013.

[135] Eiter T, Faber W, Leone N, *et al*. A logic programming approach to knowledge-state planning, II: The DLVK system. Artificial Intelligence. 2003; 144(1-2): 157-211.

[136] Mediratta A, Srivastava B. Applying planning in composition of web services with a user-driven contingent planner. IBM Research; 2006. RI 06002.

[137] Nau D, Ghallab M, Traverso P. Automated planning: Theory and practice. Morgan Kaufmann; 2004.

[138] Dacosta LAG, Pires PF, Mattoso M. Automatic composition of web services with contingency plans. Proceedings of the 2ndInternational Conference on Web Services Workshop (ICWS); 2004.

[139] Zou G, Chen Y, Xu Y, *et al*. Towards automated choreographing of web services using planning. Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI); 2012.

[140] Kautz H, Selman B, Hoffmann J. SatPlan: planning as satisfiability. Proceedings of the 5th International Planning Competition (IPC); 2006.

[141] Wang P, Ding Z, Jiang C, *et al*. Automatic Web Service Composition Based on Uncertainty Execution Effects. IEEE Transactions on Services Computing. Forthcoming 2015. http://dx.doi.org/10.1109/TSC.2015.241294