Prog Artif Intell (2017) 6:299–314
DOI 10.1007/s13748-017-0125-5

REGULAR PAPER

Incremental contingency planning for recovering from critical outcomes in high-probability seed plans

Yolanda E-Martín[1] · María D. R-Moreno[2] · David E. Smith[3]

Received: 13 February 2017 / Accepted: 4 April 2017 / Published online: 20 April 2017

Abstract Planning is the problem of choosing and organizing a sequence of actions that when applied in a given initial state results in a goal state. However, in real problems unexpected action outcomes may occur and the initial state of the world may not be known with certainty. *Incremental contingency planning* considers potential failures in a plan and attempts to avoid them by incrementally adding contingency branches to the plan in order to improve the overall probability. The planner focuses on high-probability outcomes and attempts to avoid them by incrementally adding contingency branches to the plan in order to improve the overall probability. Some of these high-probability outcomes might be repairable by runtime replanning so we focus on repairing critical outcomes that cannot be fixed by runtime replanning. For this planning to be successful, we also need high-probability seed plans. In this work, we describe approaches to generating high-probability seed plans and to incremental contingency planning on the critical outcomes.

Keywords Probabilistic planning · Plan graph propagation · Probability interaction · Heuristic search

✉ Yolanda E-Martín
yolanda.escudero@uc3m.es

María D. R-Moreno
mdolores@aut.uah.es

David E. Smith
david.smith@nasa.gov

1 Universidad Carlos III de Madrid, Av. Universidad 30, 28911 Leganés, Madrid, Spain

2 Universidad de Alcalá, Ctra Madrid-Barcelona Km 33.6, 28871 Alcalá de Henares, Madrid, Spain

3 NASA Ames Research Center, Moffett Field, CA 94035, USA

1 Introduction

Classical planning is the problem of choosing and organizing a sequence of actions that when applied in a given initial state results in a goal state. It is based on the assumption of complete knowledge of the initial state and the effects of actions. However, in real planning problems actions may have unexpected outcomes and the initial state of the world may not be known with certainty. A line of research dealing with planning problems under uncertainty is probabilistic planning, which describes the uncertainty using probability distributions.

Incremental contingency planning (ICP) is a framework that considers potential failures in a plan and attempts to avoid them by incrementally adding contingency branches to the plan in order to improve the overall probability [5]. As initially conceived, ICP focuses on high-probability outcomes. However, some of these high-probability outcomes might be repairable by runtime replanning and we could, therefore, focus on repairing critical outcomes that cannot be fixed by runtime replanning.

In this work, we present an approach to incrementally generating contingency branches to only deal with critical outcomes. The main idea is to first generate a high-probability non-branching seed plan, which is then augmented with contingency branches to handle the most critical outcomes. Any remaining outcomes are handled by runtime replanning. For the most critical outcomes, we attempt to improve the chances of recovery by (1) revising the plan to avoid or reduce the probability of getting to that outcome, (2) adding precautionary steps that allow recovery, if the failure occurs, or (3) adding a conformant path that can achieve the goal by using a different path. All three strategies can increase the overall probability of the plan. The process is repeated until (1) the resulting contingent plan achieves at least a given probability threshold, (2) the available time is exhausted, or (3) a certain number of branches are added.

In Sect. 2, we briefly describe PIPSS^I [6], a system that adopts the Determinization and Replanning approach for generating non-branching seed plans. In Sect. 3, we describe a novel approach for generating higher probability non-branching seed plans, namely *probability estimates without determinization* (PEWD), which does not rely on determinization. In Sect. 4, we define the heuristic function used to identify points of failure that potentially improve the total probability of the plan. In Sect. 5, we detail the different techniques we can apply to improve the chances of recovery, if a failure occurs. In Sect. 6, we present an empirical evaluation. Finally, in Sect. 7, we discuss the limitations of our approach and outline some future work.

2 Seed plans from all-outcomes determinization

Determinization consists of transforming a probabilistic planning domain into a deterministic planning domain. All-outcomes determinization generates a deterministic action for each outcome of a probabilistic action. Consider the probabilistic action (drive trk a b) defined in Fig. 1a with two outcomes, one where the truck arrives at its destination normally, and the other where it arrives with a flat tire. Applying all-outcomes determinization results in two deterministic actions. The most likely outcome of the action implies that the car successfully drives between locations with probability 0.6. This results in action (drive-1 trk a b), shown in Fig. 1b. For the other outcome, the car achieves the destination, but it gets a flat tire with a probability of 0.4. This results in action (drive-2 trk a b), shown in Fig. 1c.

The classical all-outcomes determinization does not make use of the probabilistic information in the domain description, which may result in frequent replanning. To overcome this issue, Jiménez et al. [9] developed an approach that turns the probability information of each outcome into an additive cost C equal to the negative logarithm of the probability. That is, $C(a') = -\ln(p_o)$. Then, they search for a plan using a numeric deterministic planner that minimizes cost. In our example, this conversion process builds the same two deterministic actions (drive-1 trk a b) and (drive-2 trk a b) with additive costs $C(\text{drive-1 trk a b}) = -\ln(0.6) = 0.51$ and $C(\text{drive-2 trk a b}) = -\ln(0.4) = 0.91$ respectively.

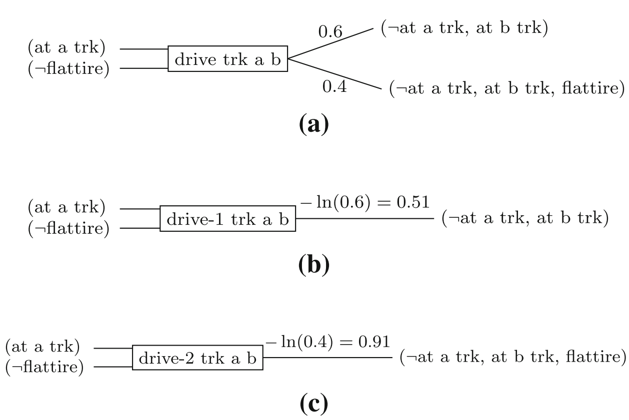

Fig. 1 All outcomes determinization considering the probability of propositions across action outcomes. **a** PPDDL action (drive trk a b). **b** Determinized PDDL action (drive-1 trk a b). **c** Determinized PDDL action (drive-2 trk a b)

Converting probability information into costs makes it possible to use deterministic numeric planners to find higher probability seed plans [9]. Following this paradigm we developed PIPSS^I [6], a forward heuristic planner that adopts the determinization and replanning approach with the aim of producing high-probability seed plans that are less likely to get stuck in dead-end states. This system initially translates the probabilistic problem into a deterministic one by using the technique of Jiménez, Coles, and Smith. Then, the system builds a plan graph for the purpose of estimating costs. The system uses this information to guide forward state-space search using A*. For each state, the plan graph is updated, and a relaxed plan is created to estimate the cost (probability) of achieving the goals from that state. This estimation is called the *completion cost estimate* (CCE).

All-outcomes determinization has proven very successful in several systems [9,11–14]. However, considering the outcomes independently can underestimate the probability of propositions. For any single outcome of an action, the probability of a proposition may be lower than considering the probability across all the outcomes. As a result, action determinization as done in PIPSS^I may mislead the planner into picking the wrong outcome or action. To illustrate this, consider the simple action A shown in Fig. 2, which has three outcomes: outcome o_1 that produces x and y with probability 0.3; outcome o_2 that produces x with probability 0.3; and outcome o_3 that produces z with probability 0.4. Outcomes o_1 and o_2 have a common proposition x, while outcome o_3 produces a different proposition that does not occur in any other outcome. Suppose that the three outcomes lead the planner to the goal with equal probability. The outcome o_3 has a probability of 0.4, which is higher than the probability of either o_1 or o_2. Therefore, a cost minimizing planner would likely choose outcome o_3. However, the true probability of x is a combination of outcomes o_1 and o_2. The combination of these outcomes will lead to a probability of 0.6 for x and, therefore, result in a better plan.

To overcome this issue, we could use a determinization approach in which we created a new deterministic action for each possible proposition combination across all of the action's outcomes. To illustrate, consider again the probabilistic action A from Fig. 2. We could create deterministic action A_1 for proposition x with probability 0.6 because x is

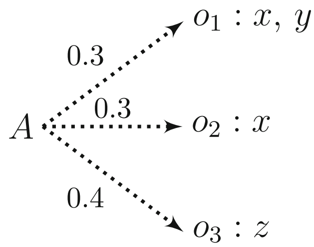

Fig. 2 Example of a probabilistic action where determinization can pick the wrong outcome

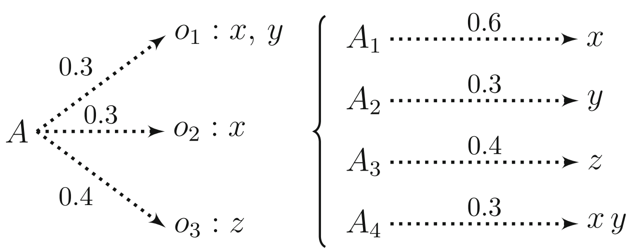

Fig. 3 Example of a potential determinization technique

in outcomes o_1 and o_2; A_2 for proposition y with probability 0.3 because y is only in outcome o_1; A_3 for proposition z with probability 0.4 because z is only in outcome o_3; and A_4 for the pair of propositions (x, y) from outcome o_1 with probability 0.3. There is no outcome that contains y and z or all three x, y, and z, so these possibilities do not need to be considered. These new deterministic actions shown in Fig. 3 would be mutually exclusive, and we can use them in probability propagation as is done in PIPSS^I [6]. However, this would significantly increase the number of actions in the plan graph and, therefore, increase propagation time.

To overcome this issue, we instead consider the overall probability of each proposition across all of the action's outcomes, and the dependence between those propositions. In the next section, we introduce a technique to compute estimates of probability without determinization. These estimates are then used to guide the search toward higher probability plans.

3 Seed plans without determinization

In this section, we present a new way to estimate probabilities that we call *probability estimates without determinization* (PEWD), which does not rely on determinization. This approach considers the probabilistic problem without transforming it into a deterministic one. Given a PPDDL problem, we initially process and load all the information given in the domain. Then, we build a probabilistic plan graph estimator as will be described in Sect. 3.3. This propagation technique is different from the standard probability propagation in plan graphs because (1) it considers the dependence among propositions in action outcomes to avoid the reliance on individual outcomes, and (2) it propagates probability rather than cost since it directly deals with probabilistic actions.

The next subsection explains the concept of probability *interaction*. Then, Sect. 3.2 describes the search in the space of probabilistic states. Finally, Sect. 3.3 describes the probabilistic plan graph heuristic used to guide the probabilistic search toward high-probability seed plans.

3.1 Probability interaction

Bryce and Smith [2] define *interaction*, I, between two elements as the probability of the conjunction divided by the individual probabilities. I, therefore, represents how more or less likely it is that two propositions or actions are established together instead of independently. Formally, the optimal interaction, I^*, considers n-ary interaction relationships among propositions and among actions in the plan graph. It is defined as:

$$I^*(p_0, p_1, \ldots, p_n) = \frac{pr^*(p_0 \wedge p_1 \wedge \cdots \wedge p_n)}{pr^*(p_0)\, pr^*(p_1) \cdots pr^*(p_n)} \tag{1}$$

where the term $pr^*(p_0 \wedge p_1 \wedge \cdots \wedge p_n)$ is the maximum probability among all the possible plans that achieve the conjunction. Computing I^* would be computationally prohibitive. As a result, we limit the calculation of these values to pairs of propositions and pairs of actions in each level of a plan graph. In other words, binary interaction is defined as:

$$I(p, q) = \frac{pr(p \wedge q)}{pr(p)\, pr(q)} \tag{2}$$

I has the following characteristics:

$$I(p, q) \text{ is } \begin{cases} >1 & \text{if } p \text{ and } q \text{ are } \textit{synergistic} \\ =1 & \text{if } p \text{ and } q \text{ are } \textit{independent} \\ <1 & \text{if } p \text{ and } q \text{ } \textit{interfere} \\ =0 & \text{if } p \text{ and } q \text{ are } \textit{mutually exclusive} \end{cases}$$

I provides information about the degree of interference or synergy between pairs of propositions and pairs of actions in a plan graph. When $0 < I(p, q) < 1$ it means that there is some interference between the best plans for achieving p and q, so it is less likely to achieve them both than to achieve them independently. In the extreme case, $I = 0$, the propositions or actions are mutually exclusive. Similarly, when $I(p, q) > 1$ the two elements are synergistic, which means that the probability of establishing both p and q is higher than the product of the probabilities for establishing the two independently. However, this probability cannot be higher than the probability of establishing the most difficult of p and q. As a result:

$$I(p, q) \leq \frac{\min\{pr(p), pr(q)\}}{pr(p)pr(q)} = \frac{1}{\max\{pr(p), pr(q)\}} \quad (3)$$

3.2 Search in the space of probabilistic states

We define a probabilistic state s as consisting of a set of propositions with individual probabilities $\mathcal{P}r(x)$ together with a probability interaction $I(x, y)$ for all pairs x and y in s.

The following subsections describe in detail how to compute the probability and interaction information in a probabilistic state.

3.2.1 Calculating probabilities for a probabilistic state

Consider a probabilistic state s and let s' be the new state after attempting to perform action a, with set of preconditions $\mathcal{P}_a$, in s. The probability of a proposition x' in s' is given by the probability of getting the proposition when the action succeeds plus the probability of getting the proposition when the action fails.[1] That is:

$$\begin{aligned}
\mathcal{P}r(x') &= pr(x'|a)\, pr(a) + pr(x'|\neg a)\, pr(\neg a) \\
&= pr(x'|a)\, pr(a) + pr(x|\neg a)\, pr(\neg a) \\
&= pr(x'|a)\, pr(a) + pr(x)\, pr(\neg a|x) \\
&= pr(x'|a)\, pr(a) + pr(x)\, (1 - pr(a|x)) \\
&= pr(x'|a)\, pr(a) + pr(x)\, (1 - pr(\mathcal{P}_a|x)) \\
&= pr(x'|a)\, pr(a) + pr(x) - pr(x)\, pr(\mathcal{P}_a|x) \\
&= \underbrace{pr(x'|a)\, pr(a)}_{T_1} + \underbrace{pr(x) - pr(x \wedge \mathcal{P}_a)}_{T_2} \quad (4)
\end{aligned}$$

The first term T_1 can be rewritten in terms of the action's outcomes as:

$$pr(x' \mid a)\, pr(a) = pr(a) \sum_{o \in \mathcal{O}(a)} pr(o)\, pr(x' \mid o, a)$$

where the conditional probability of x given an outcome o of action a is defined as:

$$pr(x|o, a) = \begin{cases} 1 & \text{if } (x \in o) \\ 0 & \text{if } (\neg x \in o) \\ pr(x|\mathcal{P}_a) & \text{if } (x, \neg x \notin o) \end{cases}$$

In other words, if the outcome o produces the proposition x, then the conditional probability is 1 (the outcome is considered). If o produces $\neg x$, then the conditional probability is 0 (the outcome is not considered). Finally, if o does not produce either x or $\neg x$, then the probability is that of x persisting through a, which depends on the probability of x given the set of preconditions $\mathcal{P}_a$ of a:

$$pr(x|\mathcal{P}_a) = \begin{cases} 1 & \text{if } (x \in \mathcal{P}_a) \\ 0 & \text{if } (\neg x \in \mathcal{P}_a) \\ pr(x) \prod_{p_i \in \mathcal{P}_a} I(x, p_i) & \text{if } (x, \neg x \notin \mathcal{P}_a) \end{cases}$$

In other words, if the proposition x belongs to the action's preconditions, the conditional probability is 1 (the outcome is considered). If $\neg x$ belongs to the action's preconditions, the conditional probability is 0 (the outcome is not considered). If x and $\neg x$ do not belong to the action's preconditions, then it is necessary to compute the probability that x holds given the preconditions of a, which is the probability of x times the interaction of x with the preconditions of a.

The second term T_2 in Eq. 4 computes the probability of the proposition assuming that the action fails to execute. The first term in T_2 refers to the probability of the proposition before the action is applied. The second term in T_2 refers to the probability that x is consistent with the set of preconditions $\mathcal{P}_a$ of a, which is given as:

$$pr(x \wedge \mathcal{P}_a) = \begin{cases} pr(a) & \text{if } (x \in \mathcal{P}_a) \\ pr(a)\, pr(x) \prod_{p \in \mathcal{P}_a} I(p, x) & \text{if } (x \notin \mathcal{P}_a) \end{cases}$$

In other words, if the proposition x belongs to the action's preconditions, the term reduces to the probability of the action. Otherwise, it is necessary to consider the interaction between x and the action's preconditions.

To illustrate, consider the planning problem shown in Fig. 4, where there is a package *pkg* and a truck *trk* at location a, and the package needs to be delivered to location c. The truck can move between different locations, and it may have a flat tire during a move with 0.4 probability. Location d has a spare tire.

Figure 5 shows the transition process from a probabilistic state S_0 to a probabilistic state S_1 for this simple problem. The probabilistic state S_0 is the initial state, where each proposition has probability equal to 1. The probabilistic state S_1 is the result of applying (drive trk a d) to S_0, where the probability

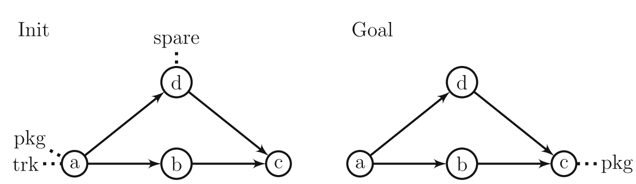

Fig. 4 Initial and goal states for a logistics problem

[1] We assume that the executive is smart enough that it will not execute an action if its preconditions are not satisfied so the state remains unchanged in this case.

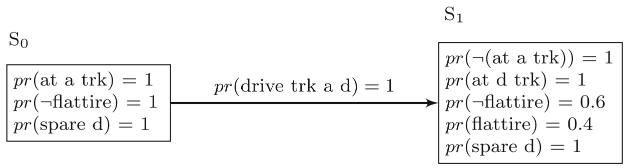

Fig. 5 Example of the transition from a probabilistic state to another probabilistic state

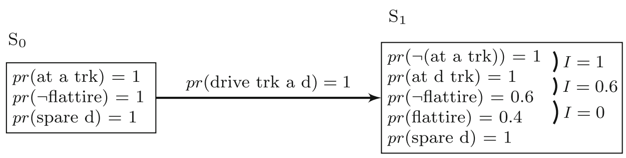

Fig. 6 Example of the transition from a probabilistic state to another probabilistic state with interaction information

of propositions (at b trk) and ¬(at a trk) is 1, the probability of ¬(flattire) is 0.6, and, therefore, the probability of (flattire) is 0.4 (Fig .6).

3.2.2 Calculating interaction information for a probabilistic state

The propositions in a probabilistic state are not independent of each other. It is, therefore, necessary to capture and store the interaction between each pair of propositions. The interaction for a pair of propositions in s' is:

$$I(x', y') = \frac{pr(x' \wedge y')}{pr(x')\, pr(y')} \quad \leq \quad \frac{1}{\max\{pr(x), p(y)\}}$$

where the conjunction probability of x' and y' is given by the probability of getting both when the action succeeds plus the probability of getting both when the action fails. That is:

$$\begin{aligned} pr(x' \wedge y') &= pr(x' \wedge y'|a)\, pr(a) + pr(x \wedge y|\neg a)\, pr(\neg a) \\ &= pr(x' \wedge y'|a)\, pr(a) + pr(x \wedge y|\neg a)\, pr(\neg a) \\ &= pr(x' \wedge y'|a)\, pr(a) + pr(x \wedge y)\, pr(\neg a|x \wedge y) \\ &= pr(x' \wedge y'|a)\, pr(a) + pr(x \wedge y)\,(1 - pr(a|x \wedge y)) \\ &= \underbrace{pr(x' \wedge y'|a)\, pr(a)}_{T_1} + \underbrace{pr(x \wedge y) - pr(x \wedge y \wedge \mathcal{P}_a)}_{T_2} \end{aligned} \tag{5}$$

As before, the term T_1 in Eq. 5 can be rewritten in terms of the action's outcomes as:

$$pr(x' \wedge y'|a)\, pr(a) \;=\; pr(a) \sum_{o \in \mathcal{O}a} pr(o)\, pr(x' \wedge y'|o, a)$$

where the conditional probability of x' and y' given an outcome o of action a, with set of preconditions $\mathcal{P}_a$, ($pr(x' \wedge y'|o, a)$) is:

- 1 if $(x, y \in o)$
- 0 if $(\neg x \in o)$ or $(\neg y \in o)$
- $pr(y|\mathcal{P}_a)$ if $(x \in o)$ and $(y, \neg y \notin o)$
- $pr(x|\mathcal{P}_a)$ if $(x, \neg x \notin o)$ and $(y \in o)$
- $pr(x \wedge y|\mathcal{P}_a)$ if $(x, \neg x, y, \neg y \notin o)$

In other words, if the outcome o produces propositions x and y, then the conditional probability is 1 (the outcome is considered). If o produces $\neg x$ or $\neg y$, then the conditional probability is 0 (the outcome is not considered). If o produces x and does not produce y or $\neg y$, then it is necessary to compute the probability that y persists through a, which depends on the probability of y given the preconditions of a. If o produces y and does not produce x or $\neg x$, then it is necessary to compute the probability that x persists through a, which depends on the probability of x given the preconditions of a. If o does not produce x, $\neg x$, y, or $\neg y$, then it is necessary to compute the probability that x and y both persist through a, which depends on the probability of x and y given the preconditions of a ($pr(x \wedge y|\mathcal{P}_a)$), which is:

- 1 if $(x, y \in \mathcal{P}_a)$
- 0 if $(\neg x \in \mathcal{P}_a)$ or $(\neg y \in \mathcal{P}_a)$
- $pr(x) \prod_{p \in \mathcal{P}_a} I(x, p)$ if $(x \notin \mathcal{P}_a)$ and $(y \in \mathcal{P}_a)$
- $pr(y) \prod_{p \in \mathcal{P}_a} I(y, p)$ if $(x \in \mathcal{P}_a)$ and $(y \notin \mathcal{P}_a)$
- $pr(x) pr(y) \prod_{p \in \mathcal{P}_a} I(x, p) I(y, p)$ if$(x, \neg x, y, \neg y \notin \mathcal{P}_a)$

The term T_2 in Eq. 5 computes the probability of the conjunction x and y assuming that the action fails to execute. The first term in T_2 refers to the conjunction probability of x and y before a is applied. The second term in T_2 refers to the probability that x and y are consistent with the preconditions of a ($pr(x \wedge y \wedge \mathcal{P}_a)$), which is given as:

- $pr(a)$ if $(x, y \in \mathcal{P}_a)$
- $pr(a)\, pr(x) \prod_{p \in \mathcal{P}_a} I(p, x)$ if $(x \notin \mathcal{P}_a)$ and $(y \in \mathcal{P}_a)$
- $pr(a)\, pr(y) \prod_{p \in \mathcal{P}_a} I(p, y)$ if $(x \in \mathcal{P}_a)$ and $(y \notin \mathcal{P}_a)$
- $pr(a)\, pr(x)\, pr(y) \prod_{p \in \mathcal{P}_a} I(p, x)\, I(p, y)$ if $(x, y \notin \mathcal{P}_a)$

Figure 9 shows again the transition process from S_0 to S_1 with probability information for each probabilistic proposition, and interaction information between some pairs of propositions at S_1. As an example, the interaction between propositions (at d trk) and ¬(flattire) at S_1 is 0.6. An interaction value of 0.6 means that there is some interference between propositions. This interference comes from the fact that one of the action's outcomes produces (flat-tire).

3.3 Probability and interaction propagation in plan graphs for PEWD

In the previous section, we described the concept of probabilistic states and how to compute them. In searching for a plan, we also need a heuristic estimate to help the planner decide what state to expand next. In order to do this, we need an estimate of how likely the state is to lead to the goals. In this section, we describe an approach to computing more accurate estimates of probability that allow the planner to search toward non-branching seed plans with high probability of success. We first describe how we do this probability estimation considering the overall probability of each proposition across all of the action's outcomes, and the dependencies between propositions in the different outcomes. Then, we describe a heuristic function that makes use of this probability estimation to guide a planner toward high probability of success plans.

As in previous work [6,7], probability and interaction information can be estimated using a plan graph. The computation of probability and interaction information begins at level zero of the plan graph where the probability of the propositions and their pairwise interactions are given by the probabilistic state.

It is important to note that the probabilities and interaction values propagated in the plan graph are approximations since the calculation of these values is limited to pairs of propositions and pairs of actions in each level of a plan graph.

3.3.1 Computing action Probability and interaction

The probability and interaction information of a proposition layer at a given level of the plan graph is used to compute the probability and the interaction information for the subsequent action layer. In particular, considering an action a at level l with a set of preconditions $\mathcal{P}_a$, the estimation of how likely it is to execute the action is the product of achieving all its preconditions times the interaction between all pairs of preconditions:

$$pr(a) \approx \prod_{x \in \mathcal{P}_a} pr(x) \prod_{\substack{(x_i, x_j) \in \mathcal{P}_a \\ j > i}} I(x_i, x_j) \tag{6}$$

where $pr(a) \leq \max_{x \in \mathcal{P}_a} pr(x)$.

The interaction between two actions a and b at level l, with sets of preconditions $\mathcal{P}_a$ and $\mathcal{P}_b$, is defined as:

$$I(a, b) = \begin{cases} 0 & \text{if } a \text{ and } b \text{ are mutex by inconsistent} \\ & \text{effects or interference} \\ \frac{pr(a \wedge b)}{pr(a)\, pr(b)} & \text{otherwise} \end{cases} \tag{7}$$

The probability of both actions $pr(a \wedge b)$ is the probability of the union of their preconditions $pr(\mathcal{P}_a \cup \mathcal{P}_b)$, which is approximated as the product of the probabilities of achieving all their preconditions times the interaction between all pairs of preconditions. That is:

$$pr(\mathcal{P}_a \cup \mathcal{P}_b) \approx \prod_{x \in \mathcal{P}_a \cup \mathcal{P}_b} pr(x) \prod_{\substack{(x_i, x_j) \in \mathcal{P}_a \cup \mathcal{P}_b \\ j > i}} I(x_i, x_j)$$

The interaction above can be simplified to:

$$I(a, b) \approx \frac{\prod_{\substack{x_i \in \mathcal{P}_a - \mathcal{P}_b \\ x_j \in \mathcal{P}_b - \mathcal{P}_a}} I(x_i, x_j)}{\prod_{x \in \mathcal{P}_a \cap \mathcal{P}_b} pr(x) \prod_{\substack{(x_i, x_j) \in \mathcal{P}_a \cap \mathcal{P}_b \\ j > i}} I(x_i, x_j)} \tag{8}$$

where the numerator is the interaction between unique preconditions for each action, and the denominator is the probability of common preconditions and the interaction between them.

To illustrate, consider a simple problem with operator A that has preconditions x, y, and t, and operator B that has preconditions x, y, and z. Assuming that A and B are not mutually exclusive, the interaction between actions A and B will be:

$$I(A, B) = \frac{pr(\mathcal{P}_A \cup \mathcal{P}_B)}{pr(A)\, pr(B)} \tag{9}$$

where:

$$pr(\mathcal{P}_A \cup \mathcal{P}_B) = pr(t)\, pr(x)\, pr(y)\, pr(z)\, I(t, x)\, I(t, y)\, I(t, z)\, I(x, y)\, I(x, z)\, I(y, z)$$

$$pr(A) = pr(t)\, pr(x)\, pr(y)\, I(t, x)\, I(t, y)\, I(x, y)$$

$$pr(B) = pr(x)\, pr(y)\, pr(z)\, I(x, y)\, I(x, z)\, I(y, z)$$

Therefore, the interaction between A and B can be simplified to:

$$I(A, B) = \frac{I(t, z)}{pr(x)\, pr(y)\, I(x, y)}$$

Figure 7 shows a partial plan graph for the Logistics problem. The numbers above the propositions and actions are the probabilities associated with each one, computed during the probability propagation process. The numbers next to

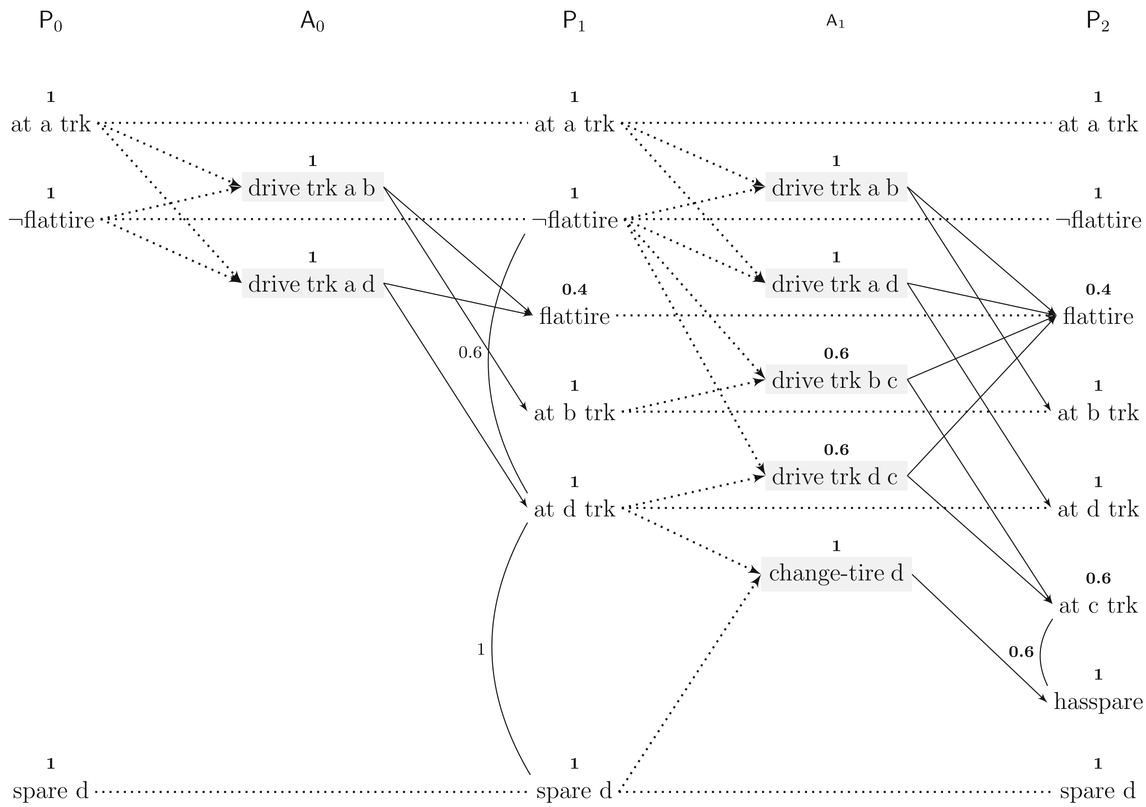

Fig. 7 A partial plan graph with probability values of propositions and actions

the edges are the interaction between the two elements connected by the edges. As an example, the probability of action (drive d c) at level 1 is 0.6, and the probability of proposition (change-tire d) at level 1 is 1.

3.3.2 Computing proposition probability and interaction

To estimate the probability of a proposition at a level, all the possible actions at the previous level that achieve that proposition need to be taken into account. We make the usual optimistic assumption that we can use the action that maximizes the probability, but we are considering the action as a whole. To do this, we must consider all outcomes of the action that contribute to the proposition. More formally, for a proposition x at level l, achieved by actions $\mathcal{A}_x$ at the preceding level, the probability is calculated as:

$$pr(x) = \max_{a \in \mathcal{A}(x)} \left\{ pr(a) \sum_{o \in \mathcal{OA}(a,x)} pr(o)\, pr(x \mid o, a) \right\} \tag{10}$$

where $\mathcal{OA}(a, x)$ is the set of outcomes of action a that produce x. Therefore, the second term in the equation gives information about the total probability of x given the action a. This information is given by the conditional probability of x given o, which is defined as:

$$pr(x \mid o, a) = \begin{cases} 1 & \text{if } (x \in o) \\ 0 & \text{if } (\neg x \in o) \\ pr(x \mid \mathcal{P}_a) & \text{if } (x, \neg x \notin o) \end{cases} \tag{11}$$

where $\mathcal{P}_a$ is the set of preconditions of a. If the outcome o produces the proposition x, then the conditional probability is 1 (the outcome is considered). If o produces $\neg x$ (deletes x), then the conditional probability is 0 (the outcome is not considered). Finally, if o does not produce either x or $\neg x$, then we need to compute the probability that x persists through the action. This requires considering the relationship between x and the action's preconditions at the previous level. If x belongs to the action's preconditions, then the conditional probability is 1 (x is necessary for the action and the outcome is considered). If $\neg x$ belongs to the action's preconditions, the conditional probability is 0 (x is inconsistent with the action so the outcome is not considered). If x or $\neg x$ do not belong to the action's preconditions, then it is necessary to consider whether the proposition was present in the previous layer given the preconditions of the action. Formally:

$$pr(x \mid \mathcal{P}_a) = \begin{cases} 1 & \text{if } (x \in \mathcal{P}_a) \\ 0 & \text{if } (\neg x \in \mathcal{P}_a) \\ pr(x) \prod_{p \in \mathcal{P}_a} I(x, p) & \text{if } (x, \neg x \notin \mathcal{P}_a) \end{cases} \tag{12}$$

Finally, we compute the interaction between propositions. In order to calculate the interaction between two propositions x and y at a level l, we need to consider all the possible ways to achieve both propositions. In other words, all the actions that achieve the pair of propositions, and the interaction between them. Suppose that $\mathcal{A}_x$ and $\mathcal{A}_y$ are the sets of actions that achieve propositions x and y, respectively, at level l. The interaction between x and y is then:

$$I(x,y) \approx \frac{\max \left\{ \begin{array}{l} \max\limits_{\substack{a \in \mathcal{A}_x \cap \mathcal{A}_y \\ a \notin \text{noop}}} pr(a)\, pr(x \wedge y | a), \\ \\ \max\limits_{\substack{a \in \mathcal{A}_x,\, b \in \mathcal{A}_y \\ a \notin \text{noop},\, b \notin \text{noop} \\ a \neq b}} pr(a \wedge b)\, pr(x \wedge y | a \wedge b), \\ \\ pr(x)\, pr(y)\, I(x,y) \end{array} \right\}}{pr(x)\, pr(y)} \tag{13}$$

The first term in the max expression corresponds to those actions that accomplish both propositions x and y. It is computed as:

$$\max_{\substack{a \in \mathcal{A}_x \cap \mathcal{A}_y \\ a \notin \text{noop}}} \{pr(a)\, pr(x \wedge y | a)\}$$
$$= \max_{\substack{a \in \mathcal{A}_x \cap \mathcal{A}_y \\ a \notin \text{noop}}} \left\{ pr(a) \sum_{o \in \mathcal{O}_a} pr(o)\, pr(x \wedge y | o, a) \right\}$$

where $\mathcal{O}_a$ is the set of outcomes of action a. The conditional probability of x and y given an outcome o ($pr(x \wedge y | o, a)$) is given as:

- 1 if $(x, y \in o)$
- 0 if $(\neg x \in o)$ and $(\neg y \in o)$
- $pr(x \,|\, \mathcal{P}_a)$ if $(y \in o)$ and $(x, \neg x \notin o)$
- $pr(y \,|\, \mathcal{P}_a)$ if $(x \in o)$ and $(y, \neg y \notin o)$
- $pr(x \wedge y \,|\, \mathcal{P}_a)$ if $(x, \neg x, y, \neg y \notin o)$

Similarly, the second term in the max expression corresponds to those actions that accomplish only one proposition each. It is given as:

$$\max_{\substack{a \in \mathcal{A}_x,\, b \in \mathcal{A}_y \\ a \notin \text{noop},\, b \notin \text{noop}}} \{pr(a \wedge b) pr(x \wedge y | a \wedge b)\}$$

which is equal to:

$$\max_{\substack{a \in \mathcal{A}_x,\, b \in \mathcal{A}_y \\ a \notin \text{noop},\, b \notin \text{noop}}} \left\{ \begin{array}{l} pr(a \wedge b) \sum\limits_{o_i \in \mathcal{O}_a} pr(o_i)\, pr(x | o_i, a, b) \\ \qquad \sum\limits_{o_j \in \mathcal{O}_b} pr(o_j)\, pr(y | o_j, a, b) \end{array} \right\}$$

where $\mathcal{O}_a$ is the set of outcomes of action a, and $\mathcal{O}_b$ is the set of outcomes of action b. The conditional probabilities $pr(x|o_i, a, b)$ and $pr(y|o_j, a, b)$ are given as:

$$pr(x|o,a,b) = \begin{cases} 1 & \text{if } (x \in o) \\ 0 & \text{if } (\neg x \in o) \\ pr(x \,|\, \mathcal{P}_a \wedge \mathcal{P}_b) & \text{if } (x, \neg x \notin o) \end{cases}$$

In other words, if o produces x, the probability is 1 (the outcome is considered). If o produces $\neg x$, the probability is 0 (the outcome is not considered). If o does not produces x and $\neg x$, then the probability is the probability that x persists through a and b, which depends on the probability of x before a given the preconditions of both a and b, which is:

$$pr(x \,|\, \mathcal{P}_a \wedge \mathcal{P}_b) = \begin{cases} 1 & \text{if } (x \in \mathcal{P}_a \cup \mathcal{P}_b) \\ 0 & \text{if } (\neg x \in \mathcal{P}_a \cup \mathcal{P}_b) \\ pr(x) \prod\limits_{p \in \mathcal{P}_a \cup \mathcal{P}_b} I(x,p) & \text{if } (x, \neg x \notin \mathcal{P}_a \cup \mathcal{P}_b) \end{cases}$$

Finally, the third term in the max expression corresponds to the case where both propositions persist through noops from the previous level. This is given as the product of the probability of each individual proposition at the previous level and the interaction between them.

Returning to the current example, the calculation of the interaction between propositions (at c trk) and (¬flattire) at level 2 is 0.6, which means that there is interference between having the package at location c and not having a flat tire. This comes from both the facts that action (drive d c) has (¬flattire) as a precondition and it has nonzero probability at level 1, and has (flattire) as an effect.

3.4 Upper bounds on probability and interaction

Because the probabilities in Eqs. 6 and 10 are estimated based on binary interaction, the resulting calculations can sometimes overestimate probability and interaction. As we previously noted in Sect. 3.1, the interaction between x and y is bounded above by:

$$I(x,y) \leq \frac{1}{\max\{pr(x), pr(y)\}} \tag{14}$$

We also noted that the probability of an action is bounded above by the minimum probability of its preconditions. That is:

$$pr(a) \leq \min_{x \in \mathcal{P}_a} pr(x) \tag{15}$$

We use these bounds at all stages of the calculation in order to help avoid overestimation.

3.5 Probabilistic heuristic estimator

Using Eqs. 6, 7, 10, and 13 we can build a plan graph and propagate probability and interaction information. The construction process finishes when two consecutive proposition layers are identical and there is quiescence in probability and interaction for all propositions and actions in the plan graph. On completion, each possible goal proposition has an estimated probability of being achieved, and there is an interaction estimation between each pair of goal propositions. Therefore, using the probability and interaction information computed in the probabilistic plan graph we can compute an estimated probability of achieving a (possibly conjunctive) goal $G = \{g_1, \ldots, g_n\}$ from a particular state n, which we call the *Completion Probability Estimate* (CPE):

$$\mathrm{CPE}(n) \approx \prod_{g \in G} pr(g) \prod_{\substack{(g_i, g_j) \in G \\ j > i}} I(g_i, g_j) \ \leq \ \min_{g \in G} pr(g) \tag{16}$$

Figure 8 shows the high-level algorithm for computing the CPE used to compute the probability estimation of reaching the goal from a particular state, which may be summarized in the following steps:

1. For each proposition p in the probabilistic state S compute the probability of p in S using Eq. 4.
2. For each each pair of propositions p and q in the probabilistic state S compute the interaction between p and q in S using Eq. 5.
3. Initialize the probabilistic plan graph with the probability and interaction information of the current state and compute the new probability and interaction estimates using Eqs. 6, 7, 10, and 13.
4. Compute the CPE of the current state S by estimating the probability of G from the Probability and interaction estimates in the updated probabilistic plan graph using Eq. 16.

Function PROBABILITYESTIMATE (s)

s	$\equiv$	the current probabilistic state
p	$\equiv$	a proposition $p \in s$
q	$\equiv$	a proposition $q \in s$
G	$\equiv$	the set of goals
g	$\equiv$	a goal proposition
CPE	$\equiv$	the completion probability estimate

1. for each $p \in s$
   $pr_s(p) \leftarrow$ COMPUTEPROBABILITY(p)
2. for each $(p, q) \in s$
   $I_{p_s}(p, q) \leftarrow$ COMPUTEINTERACTION(p, q)
3. UPDATEPRPLANGRAPH(s)
4. CPE$(s) \leftarrow \prod_{g \in G} pr(g)$
5. return CPE

Fig. 8 The CPE calculation pseudo-algorithm

3.6 An extended example

Consider the progress of the probabilistic search process shown in Fig. 9 that finds a path for the logistics problem in Fig. 4. S_0 is the initial state. Actions (drive trk a b) and (drive trk a d) are the applicable actions in S_0, and generate the probabilistic states S_1 and S_2 respectively. The path to the goal through (drive trk a d) and state S_2 has a higher probability than the path through (drive trk a b) and state S_1 because of the fact that location d has a spare tire, while location b does not. The CPE value for states S_1 and S_2 are 0.6 and 1, respectively. Therefore, the next node to be expanded is S_2 where (drive trk d c) and (change d) are the applicable actions, and generate states S_3 and S_4 respectively. The path to the goal through (change d) has a higher probability than the path through (drive trk d c). The fact that $pr(\neg\text{flattire}) = 0.6$ at S_2 lowers the probability of (drive trk d c). On the other hand, the spare tire at location d increases the probability of (change d). The CPE values for states S_3 and S_4 are 0.6 and 1 respectively. Therefore, the next node to be expanded is S_4 where (drive d c) is the applicable action, and generates state S_5. It is important to note that $pr(\neg\text{flattire})$ in S_4 increases from 0.6 to 1 after applying (change). Therefore, pr(drive d c) also increases to 1. The new state S_5 contains the goal state with probability 1 so it is, therefore, not necessary to explore the search space further. For this particular problem, our heuristic leads to a maximum search probability, and finds the following plan solution with the highest probability of success:

π = {(drive trk a d) (change d) (drive d c)}

4 Recognizing outcomes

Using the technique described in Sect. 3, we can generate high probability seed plans. We perform a forward state-space search using A* over the space of probabilistic states as is described in Sect. 3.2. We guide this search using the CPE estimate as described in the previous section.

Once a seed plan has been generated, we analyze the potential unexpected outcomes to estimate how much probability could be gained by improving the chances of recovery for that outcome. We call this estimation *Gain* and it is the maximum probability that the plan could potentially be improved by repairing the outcome. To compute Gain we can

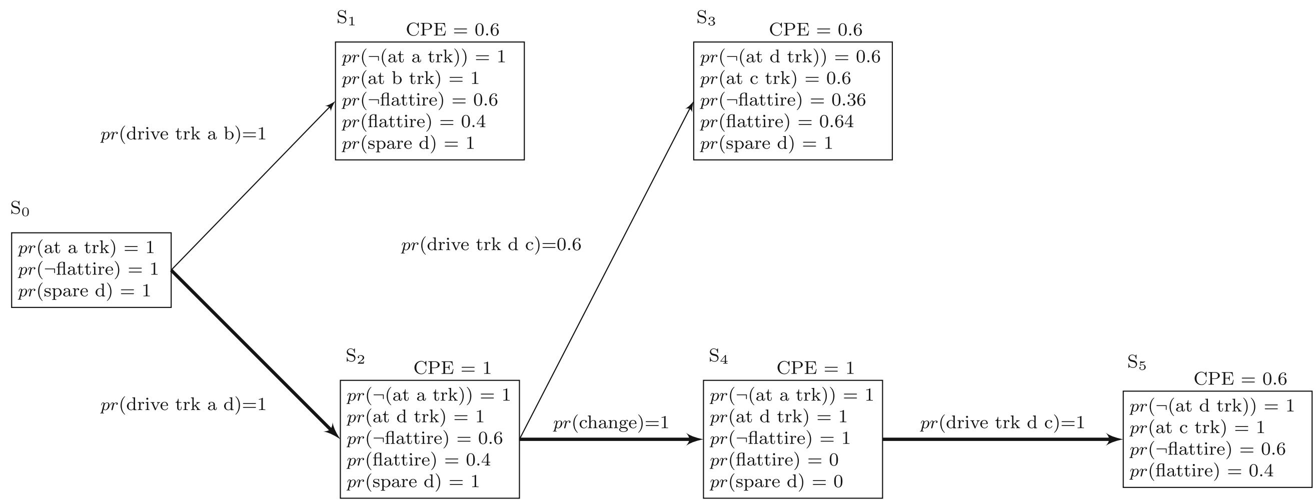

Fig. 9 Search progress using PEWD solving the Logistics problem

again use the CPE, which is used to compute an estimate of the probability of reaching the goal from that state.

For an alternative outcome (or branch) x of action a, the optimistic possible Gain from improving the branch will be the difference between the *estimated reward with repair* and the *estimated reward without repair*. We compute the latter using the CPE estimation. That is, the probability of reaching the goal from that state. To compute the *estimated reward with repair*, we propagate probability and interaction in the plan graph only considering the outcome x, but allowing other actions in the plan to change. By doing this, we force x to be in the plan and, therefore, the new probability and interaction information can be used to compute the probability of reaching the goal from that state without repair. We call this estimation *optimistic probability estimation* (OPE). More formally, for a branch x, the Gain is a measure of how much the total plan probability could potentially be increased by incremental contingency planning and is computed by the difference between the OPE of branch x and the CPE of x:

$$\text{Gain}(x) = \text{OPE}(x) - \text{CPE}(x) \tag{17}$$

To illustrate, consider the seed plan in Fig. 10. Action (drive trk a b) has an alternative outcome o_1 with probability 0.4 and CPE $= 0$. This means that there is no chance of completing the objective if this outcome actually happens—the tire goes flat and the truck cannot reach the goal. Action (drive trk b c) has an alternative outcome, o_2, with probability 0.4 and CPE $= 1$ because even though the tire goes flat, the truck still arrives at location c, and the remainder of the plan succeeds.

The Gain for branches o_1 and o_2 are:

$$\text{Gain}(o_1) = \text{OPE}(o_1) - \text{CPE}(o_1) = 0.36 - 0 = 0.36$$
$$\text{Gain}(o_2) = \text{OPE}(o_2) - \text{CPE}(o_2) = 0.36 - 1 = -0.64$$

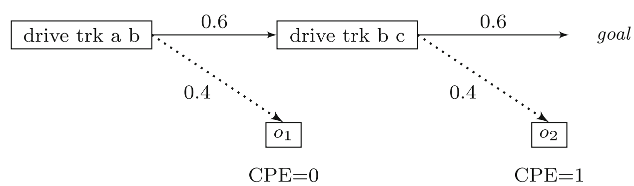

Fig. 10 Example of a non-branching seed plan with potential outcomes to be repaired

This means that by repairing branch o_1, the total plan probability will improve more than through branch o_2. Therefore, we would prefer to recover o_1 since it seems that it is possible to gain more probability mass, whereas o_2 might be recoverable by using runtime replanning. These calculations of Gain allow creating a ranking on the alternative outcomes.

5 Repairing outcomes

Given the ranking of alternative outcomes, the next step is to repair the plan in order to increase the overall probability of success. For each outcome, the idea is to look for the best improvement. In the next subsections, we present three methods to do that. The first method is called *confrontation*, which tries to find a plan that avoids the problematic action outcome. The second method is called *precautionary steps*, which adds additional (precautionary) actions before the problematic action to increase the probability of recovery in case the bad outcome happens. The third method is called *conformant augmentation*, which increases the total probability by adding conformant steps to the plan.

5.1 Confrontation

A probabilistic outcome of an action may be subject to different conditions. In our example, it might be that for the action

Function CONFRONTATION (a,o,p)

a	$\equiv$	action causing the failure
c	$\equiv$	condition on the unrecoverable outcome of a occurs
o	$\equiv$	problem operators set
p	$\equiv$	PDDL problem espacification
g	$\equiv$	set of goals
$plan$	$\equiv$	new plan solution

$a' \leftarrow \text{copy}(a)$
$prec(a') \leftarrow prec(a') \cup (\neg c)$
$eff(a') \leftarrow eff(a') \cup (\text{unique-effect})$
$o \leftarrow \{o\} \cup a'$
$g \leftarrow \{g\} \cup (\text{unique-effect})$
$plan \leftarrow \text{deterministicPlanner}(o,\ p)$
return $plan$

Fig. 11 The confrontation pseudo-algorithm

```
(:action unload
 :parameters (?pkg - package ?t - truck ?l - location)
 :precondition (and (at ?l ?t) (scanned ?pkg ?t))
 :effect (and (not (in ?pkg ?t)) (at ?l ?pkg)
          (when (and (not-broken d) (delivered pkg c)))))
```

```
(:action unload'
 :parameters (?pkg - package ?t - truck ?l - location)
 :precondition (and (at ?l ?t) (scanned ?pkg ?t) (not (not-broken d)))
 :effect (and (not (in ?pkg ?t)) (at ?l ?pkg) (delivered pkg c)
         (unique-effect)))
```

```
(define (problem logistics-p01)
 (:domain logistics)
 (:objects a b c d e - location trk - truck pkg - package
 (:init (connected a b) (connected a d) (connected b c) (connected d e)
        (connected e c) (at a trk) (at a pkg) (spare d) (spare e)
        (not (flattire)))
 (:goal (and (delivered pkg c) (unique-effect))))
```

Fig. 12 Confrontation: new action and new problem definitions

(unload pkg trk c), proposition ¬(at c pkg) occurs when, for instance, the dolly used to unload the package from the truck is broken. Confrontation on this condition will avoid ¬(at c pkg) by ensuring that the dolly is intact before the start of driving. Figure 11 shows the high-level algorithm used.

The idea is to find a new plan that avoids or reduces the probability of getting to that branch, and then replace the old seed plan with the new plan. More precisely, suppose that a is the action in the seed plan with an unrecoverable outcome conditioned by c. We force the planner to find a new seed plan that achieves $\neg c$ to prevent the failure from occurring. The way we do this is by creating a new version a' of the action a that keeps its original preconditions but adds a new additional precondition $\neg c$, and keeps its original effects but adds an additional unique effect. The unique effect is added to the set of goals. We then add the new action to the set of operators and call the deterministic planner to find a plan for the goals to force that action into the plan. If a new plan is found and it has higher probability than the old seed plan, the new plan replaces the old seed plan.

In our example, suppose that the package *pkg* needs to be delivered at location c. The action of delivering the package, *unload*, has a conditional effect (not-broken), which will deliver the package if the dolly d is intact. Figure 12 shows the new action (unload'). It includes ¬(not-broken d), the negation of the conditional effect, in its preconditions, and the proposition (unique-effect) in its effects. The new problem includes the proposition (unique-effect) in the goal set. If a solution is possible for this new problem, the deterministic planner would return a plan with action (unload') in it to guarantee that the dolly is intact before the start of the driving.

5.2 Precautionary steps

Adding precautionary steps consists of repairing an undesirable action's outcome by adding precautionary actions to the plan before the problematic action. For example, picking up a spare tire before driving in case you have a flat tire. This method improves the chance of recovery if the seed plan fails, and makes it possible to reach the goal when the unexpected outcome of the problematic action happens. Figure 13 shows the high-level algorithm used. The idea is to force the planner to find a plan that facilitates recovery from the problematic outcome, but does not lose any precondition needed to reach the goal when the action has the desired outcome. To do this, for a problematic outcome of action a we:

1. Divide the initial seed plan into two parts: a *prefix*, which contains all actions preceding a, and a *suffix*, which contains all actions following a.
2. Create a new action a' that keeps its original preconditions and effects, but adds a new effect (unique effect).
3. Analyze the causal structure of the suffix to collect all the preconditions needed by the suffix, but not added by the problematic outcome. We add these to the set of preconditions of a'.
4. Add the predicate (unique effect) to the goal state to force a' into the plan.

Function PRECAUTIONARYSTEP (a,o,p)

a	$\equiv$	action causing the failure
o	$\equiv$	problem operators set
p	$\equiv$	PDDL problem especification
g	$\equiv$	set of goals
suffix	$\equiv$	plan containing the action's seed plan following a
newPlan	$\equiv$	precautionary plan
plan	$\equiv$	plan solution

$a' \leftarrow \text{copy}(a)$
$preconditions(a') \leftarrow preconditions(a) \cup \text{causalStructure}(suffix)$
$effects(a') \leftarrow effects(a) \cup unique\text{-}effect$
$o \leftarrow \{o\} \cup a'$
$g \leftarrow \{g\} \cup unique\text{-}effect$
$newPlan \leftarrow \text{deterministicPlanner}(o,p)$
$plan \leftarrow \text{addBranch}(newPlan, suffix)$
return $plan$

Fig. 13 The precautionary steps pseudo-algorithm

Fig. 14 Precautionary steps: new action and new problem definitions

```
(:action drive'
  :parameters (?from - location ?to - location ?p - person)
  :precondition (and (at ?from) (road ?from ?to) (not (flattire)))
  :effect (and (at ?to) (not (at ?from)) (flattire) (unique-effect)))

(define (problem logistics-p01)
 (:domain logistics)
 (:objects a b c d e - location trk - truck pkg - package
 (:init (connected a b) (connected a d) (connected b c)
        (connected d e) (connected e c) (at a trk) (at a pkg)
        (spare d) (spare e) (not (flattire)))
 (:goal (and (at c pkg) (unique-effect))))
```

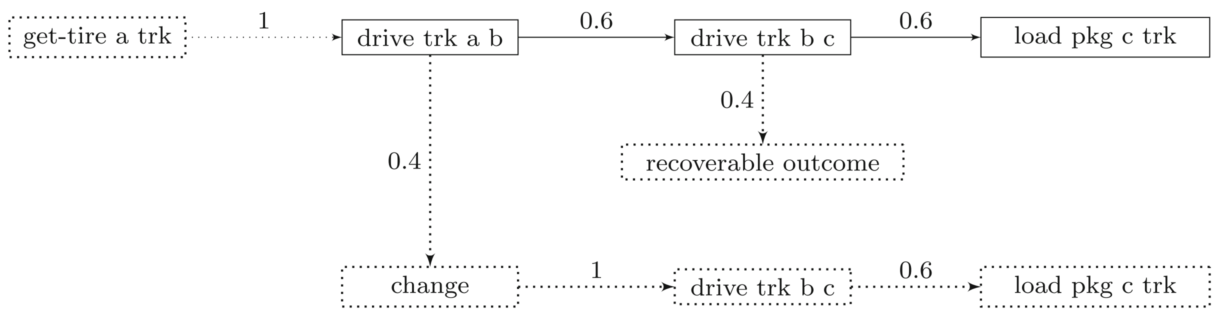

Fig. 15 Recovering action outcomes by adding a precautionary step for alternative outcome o_1

5. Add a' to the set of operators and call the deterministic planner to find a plan for the new goal state. If a plan is found and the overall probability of the plan is higher, the prefix of the seed plan is replaced with the prefix of the new plan, and the suffix is added to it as a branch for the problematic outcome of a.

Returning to our example, assume that we are repairing outcome o_1. Figure 14 shows the new action created to repair o_1. It includes the proposition (unique-effect) in its effects. Its precondition set remains the same because it already has all the preconditions necessary to enable the suffix. In addition, the new problem definition includes the proposition (unique-effect) in the goal set. The deterministic planner returns a new plan that has the precautionary action (get-tire), which increases chances of recovery in case the unexpected outcome o_1 occurs.

Figure 15 shows the contingency plan once outcome o_1 has been repaired by replacing the prefix with the new one that includes the action (get-tire), and a contingency branch where the tire is changed if the outcome o_1 happens and the car gets a flat tire. On the other hand, o_2 does not need to be repaired since it can be handled by runtime replanning.

5.3 Conformant augmentation

It is possible that there are several plans that reach the goal, which are not initially generated because they have lower probability. In some cases, one or more of these plans may be concurrently executable with the original seed plan and will raise the probability of the plan. Conformant plans may be generated when the precautionary steps method is applied. This is the case when the plan that is generated contains action *a'* (the one forced to be in the plan), but it is only in the plan to achieve the unique effect.

As an example of this technique, consider the unrecoverable outcome of action (drive trk b c) shown in Fig. 16. We can increase the overall probability of reaching the goal by simultaneously sending a second truck *trk2* to pick the package up. During execution time, both sequences would be executed concurrently. However, since the conformant plan generated might interfere with actions in the tail of the contingency plan, we need to find all the potential *execution conditions* and consider them during the execution of the plan. An execution condition is a proposition that determines which plan continues to execute. If the execution condition is true, then the execution continues with the contingency plan. Otherwise, the execution continues with the conformant plan. In our example, only one of the trucks can pick the package up at location c. Therefore, during execution time, we need to consider the execution condition (at c trk), to disable either the conformant plan, if the proposition becomes true, or the contingency plan, if the proposition becomes false.

It may happen that the resulting conformant plan requires revision to the augmented seed plan in order to be compatible with the seed plan. This revised seed plan may have lower probability than the original seed plan. This is the case where, for instance, the truck $trk2$ in the conformant plan is a large truck that requires a driver with a specific license. The logistics company only has one driver with that license, and he

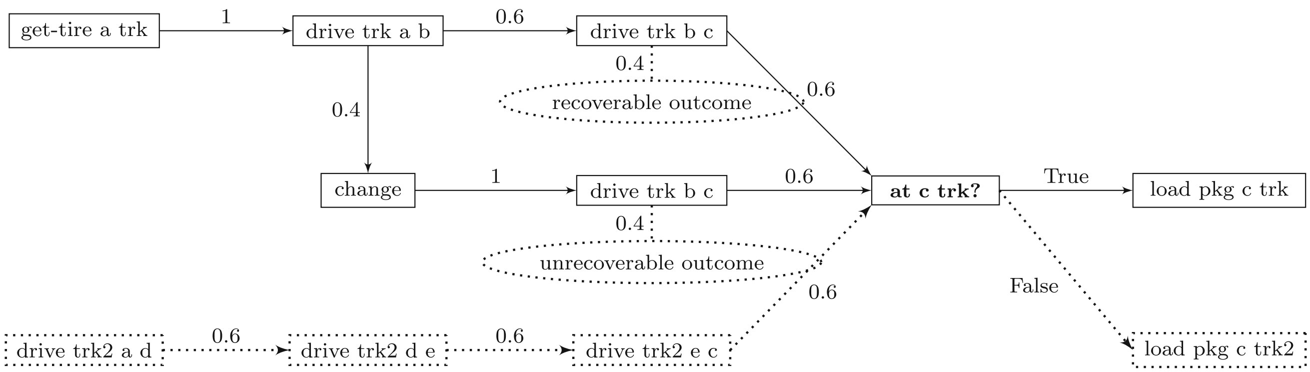

Fig. 16 Recovering action outcomes by adding a conformant plan for the unrecoverable outcome of action (drive trk b c)

was first assigned to drive truck trk. As a consequence, the revised suffix would require (1) assign that driver to $trk2$ and (2) assign a new driver to trk. The actions in the revised suffix may have some new probability of failure (for instance, the driver gets sick and cannot drive), and as a consequence of that, the overall probability of the seed plan may decrease. If the total probability of the revised seed plan plus the new conformant branch is higher than the original seed plan, then the original seed plan is replaced by the new plan with the conformant augmentation.

6 Experimental evaluation

We conducted experiments on IPPC-06 [1] and IPPC-08 [4] fully observable probabilistic planning domains, as well as on the *probabilistically interesting domains* (PID) [10]. The tests consisted of running the planner and using the resulting plan in the MDP Simulator [15]. The planner and the simulator communicate by exchanging messages. The simulator first sends the planner the initial state. Then, the interaction between planner and simulation consists of the planner sending an action and the simulator sending the next state to the planner.

The planners used for this test were FPG [3], FF-Replan [12], FHH [13], FHH$^+$ [14], and RFF [11]. We compare these with four variants of our planner:

– PIPSS$_r^I$ [6]: a planner that uses all-outcomes determinization together with probability and interaction information (turned into costs) to generate a seed plan. It does runtime replanning to deal with unexpected states at execution time.
– C-PIPSS$_r^I$ [7]: a modified PIPSS$_r^I$ planner that incrementally augments the plan solution using confrontation, precautionary steps, and conformant augmentation. It also does runtime replanning to deal with unexpected states at execution time.
– PIPSS^{IP}: a planner that uses PEWD rather than action determinization to generate a high-probability seed plan. During execution, the planner does not perform any further action when an unexpected state occurs.
– PIPSS$_r^{IP}$: a planner that uses PEWD rather than action determinization to generate a high-probability seed plan and does runtime replanning to deal with unexpected states at execution time.

The experiments were conducted on a Pentium dual-core processor at 2.4 GHz running Linux. For the rest of the planners, given that we were not able to obtain and run them ourselves, data are collected from work done by Yoon et al. [14]. To perform this test, we have chosen the Exploding-Blocksworld, Triangle-tireworld, Tireworld, Climb, and River domains because all have the property that simple replanning fails because some of the actions' outcomes yield dead-end states. (Thus, we can evaluate if our novel PEWD approach is guiding the search toward higher probability of success plans.)

Table 1 shows the number of successful rounds for FFH, FFH$^+$, FPG, PIPSS$_r^I$, C-PIPSS$_r^I$, PIPSS^{IP}, and PIPSS$_r^{IP}$ planners in each domain. For all the planners, 30 trials per problem were performed with a total limit of 30 min for the 30 trials. Exploding-Blocksworld-06, Exploding-Blocksworld-08, and Tireworld have 15 problems for each domain. So, the maximum number of successful rounds for each domain is $15 \times 30 = 450$. Triangle-tireworld only has 10 problems so that the total rounds in this case is $10 \times 30 = 300$. Climb, River, Tire1, and Tire10 have one problem for each domain, so the maximum number of successful rounds for each domain is 30.

For the Exploding-Blocksworld-06 domain, C-PIPSS$_r^I$ gets the highest rate of successful rounds closely followed by FFH$^+$, PIPSS$_r^I$, FFH, FPG, and finally PIPSS^{IP} and PIPSS$_r^{IP}$. There is the same trend for the Exploding-Blocksworld-08 domain, where FFH$^+$ stands out against the rest of the planners, followed by PIPSS$_r^I$, C-PIPSS$_r^I$, FFH, PIPSS^{IP}, and PIPSS$_r^{IP}$. The planner with the lowest rate of

Table 1 Total number of successful rounds for different planners

Domains	Planners						
	FFH	FFH^+	FPG	$PIPSS_r^I$	C-$PIPSS_r^I$	$PIPSS^{IP}$	$PIPSS_r^{IP}$
Exploding-BW-06	205	265	193	239	**266**	132	158
Tirewld-06	343	364	337	360	362	352	**365**
Climb	**30**	**30**	**30**	**30**	**30**	**30**	**30**
River	20	20	20	**23**	21	18	20
Tire1	**30**	**30**	**30**	21	18	**30**	**30**
Tire10	6	**30**	0	0	0	0	0
Total	624	**739**	610	663	697	562	603
	FFH	FFH^+	RFF	$PIPSS_r^I$	C-$PIPSS_r^I$	$PIPSS^{IP}$	$PIPSS_r^{IP}$
Exploding-BW-08	131	**214**	58	171	170	85	103
Triangle-Tirewld-08	420	420	382	21	67	210	210
Total	551	**634**	440	192	237	295	313

Bold values represent the highest success rate in each domain

Table 2 Total number of successful rounds using PEWD given unlimited amount of time

Domains	Planners			
	$PIPSS^{IP}$	$PIPSS_r^{IP}$	$u$$PIPSS^{IP}$	$u$$PIPSS_r^{IP}$
Exploding-BW-06	132	158	180	180
Exploding-BW-08	85	103	156	161
Tirewld-06	352	365	391	423
Triangle-Tirewld-08	210	210	300	300
Total	779	836	1027	1064

successful rounds is RFF, the competition winner. For the Tireworld domain, all the approaches have a similar number of successful rounds, but $PIPSS^{IP}$ has the highest rate. For Triangle-Tireworld, FFH^+ and FFH have the highest rate followed by RFF. $PIPSS^{IP}$ and $PIPSS_r^{IP}$ perform much better than $PIPSS_r^I$ and C-$PIPSS_r^I$. This is because PEWD is finding plans that avoid dead-end states, and thus manages to solve more rounds. Climb, River, Tire1, and Tire10 are problems with dead-ends and a small likelihood of simple paths. All the approaches solve all the rounds for the Climb domain. For the River domain, $PIPSS_r^I$ achieves the highest rate of successful rounds, but the other approaches are very close. For the Tire1 domain, all the approaches solve all the problems, except $PIPSS_r^I$ and C-$PIPSS_r^I$. This shows that PEWD is finding high probability of success plans. For the Tire10 domain, FFH and FHH^+ are the only planners that solve the problem and are able to complete 6 and 30 rounds respectively. (The family of $PIPSS_r^{I*}$ planners run out of time due to the size of the problem.)

With regard to the difference in performance between $PIPSS^{IP}$ and $PIPSS_r^{IP}$, the success rate is only slightly higher in $PIPSS_r^{IP}$, which performs replanning while $PIPSS^{IP}$ does not. This means that runtime replanning does not make a big difference because the technique is generating high probability of success seed plans.

It appears that $PIPSS_r^I$ and C-$PIPSS_r^I$ perform much better than $PIPSS^{IP}$ and $PIPSS_r^{IP}$ in most of the domains. The issue here is that $PIPSS_r^I$ and C-$PIPSS_r^I$ scale much better that $PIPSS^{IP}$ and $PIPSS_r^{IP}$ in term of the amount of time taken to solve the problem. $PIPSS^{IP}$ and $PIPSS_r^{IP}$ were unable to solve all the problems for the hardest domains such as Blocksworld and Tire because they run out of time due to the complexity caused by the update of the plan graph for each probabilistic state. In particular, for the Exploding-Blocksworld-06, $PIPSS^{IP}$ solves only 40% of the problems, while $PIPSS_r^I$ solves 66% of them. For this reason, the number of successful rounds for $PIPSS^{IP}$ and $PIPSS_r^{IP}$ is lower than for $PIPSS_r^I$ and C-$PIPSS_r^I$. For the Tireworld-06 domain, $PIPSS^{IP}$ solves 86% of the problems, while $PIPSS_r^I$ solves all of them. However, it still gets a high number of successful rounds, which is evidence that we are generating high probability of success plans. For Triangle-Tireworld-08, $PIPSS^{IP}$ and $PIPSS_r^{IP}$ only solve 46% of the problems, compared to $PIPSS_r^I$ and C-$PIPSS_r^I$ that solve 66% of them.

In order to confirm that this is a problem of efficiency and, therefore, the PEWD technique is generating high probability of success plans, we gave $PIPSS^{IP}$ and $PIPSS_r^{IP}$ an unlimited amount of time to solve problems for the Blocksword and Tireworld domains. Table 2 shows the results of this test. We compare the number of successful rounds for PIPSS^{IP} and PIPSS$_r^{IP}$, given 30 min, against the number of successful rounds for their counterparts uPIPSS^{IP} and uPIPSS$_r^{IP}$ respectively, given unlimited time.

For the Exploding-Blocksworld-06 domain, uPIPSS^I and uPIPSS$_r^I$ are able to solve almost 60% of the problems versus 40% given 30 min. The remainder of the problems are not solved because uPIPSS^I and uPIPSS$_r^I$ run out of memory. Despite this, the number of successful rounds increases significantly for both uPIPSS^I and uPIPSS$_r^I$. For the Exploding-Blocksworld-08 domain, uPIPSS^I and uPIPSS$_r^I$ are able to solve almost 66% of the problems versus 46% given 30 min. Again, the remainder of the problems are not solved because uPIPSS^I and uPIPSS$_r^I$ run out of memory, and the number of successful rounds increases for both uPIPSS^I and uPIPSS$_r^I$. For the Tireworld domain, uPIPSS^{IP} and uPIPSS$_r^{IP}$ solve all the problems. As a consequence, the number of successful rounds increases considerably. In particular, uPIPSS$_r^{IP}$ gets 423 of 450 successful rounds. For Triangle-Tireworld, uPIPSS^{IP} and uPIPSS$_r^{IP}$ solve all the problems and have the highest number of successful rounds. All of this suggests that PEWD is finding plans that avoid dead-end states, and its performance could be dramatically improved by improving the efficiency of the PEWD computation.

We have also tried using the PEWD technique together with the addition of incremental contingency branches. However, the efficiency of the PEWD computation is the dominant factor in determining the number of problems solved, and therefore the resulting success rate. With PEWD, if a problem can be solved within the allowed time, the success rate for the resulting seed plan is often high, and the insertion of contingency branches seems to have little additional benefit. As a result, improving the efficiency of PEWD has much greater payoff than incrementally adding contingency branches.

7 Conclusions

This work goes beyond what Foss et. al. [8] did by computing a high-probability seed plan and a *Gain* value that evaluates which outcomes will improve the overall seed plan probability. In addition, we included the *Confrontation* technique to repair outcomes subject to a condition. In general, incremental contingency planning provides little additional benefit using all-outcomes determinization for finding the seed plan. In a few domains, incremental contingency planning can help; the success rates are higher, which means that the planner has been able to reach the goal in a larger percentage of problems. However, we expected that the combination of incremental contingency planning and runtime replanning would increase the success rate for all the tested domains. Our hypothesis for the poor performance of our framework was the classical all-outcomes determinization approach. For this reason, we investigated a new way to compute estimates of probability without action determinization for probabilistic planning. This technique uses the PPDDL action definitions as is and performs search in the space of probabilistic states. The probability information provided in the domain definition is used to propagate probability and interaction information through a plan graph. This propagation technique considers the overall probability of each proposition across all of the action's outcomes and the dependencies between those propositions in the different outcomes. The resulting probabilities are then used to compute a heuristic function that guides the search toward high probability of success plans. The resulting plans are used in a system that handles unexpected outcomes by runtime replanning.

According to the results, the approach suffers from poor scalability for large domains. However, the approach has high success rates considering the number of solved problems. This is evidence that we are generating higher probability of success plans and the technique holds promise. More effort is clearly required to improve efficiency and memory usage of the probabilistic plan graph computation in order to improve scalability.

Acknowledgements This work was supported by the NASA Safe Autonomous Systems Operations (SASO) project, the MINECO project EphemeCH TIN2014-56494-C4-4-P, and the UAH project 2016/00351/001.

References

1. Bonet, B., Given, R.: International Probabilistic Planning Competition. http://www.ldc.usb.ve/~bonet/ipc5, (2006)
2. Bryce, D., Smith, D.E.: Using interaction to compute better probability estimates in plan graphs. In: Proceedings of the ICAPS-06 Workshop on Planning Under Uncertainty and Execution Control for Autonomous Systems, The English Lake District, Cumbria, UK, (2006)
3. Buffet, O., Aberdeen, D.: The factored policy-gradient planner. Artif. Intell. **173**(5–6), 722–747 (2009)
4. Buffet, O., Bryce, D.: International Probabilistic Planning Competition. http://ippc-2008.loria.fr/wiki/index.php/Main_Page, (2008)
5. Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D.E., Washington, R.: Incremental contingency planning. In: Proceedings of ICAPS-03 Workshop on Planning under Uncertainty, Trento, Italy, (2003)
6. E-Martín, Y., R-Moreno, M.D., Smith, D.E.: Progressive heuristic search for probabilistic planing based on interaction estimates. Expert Syst. **31**(5), 421–436 (2014)
7. E-Martín, Y., R-Moreno, M.D., Smith, D.E.: Incremental contingency planning for recovering from uncertain outcomes. In: Proceedings of the Conference of the Spanish Association for Artificial Intelligence, Salamanca, Spain, (2016)
8. Foss, J., Onder, N., Smith, D.E.: Preventing unrecoverable failures through precautionary planning. In: Proceedings of the ICAPS'07

Workshop on Moving Planning and Scheduling Systems into the Real World, Providence, RI, USA, (2007)
9. Jiménez, S., Coles, A., Smith, A.: Planning in probabilistic domains using a deterministic numeric planner. In: Proceedings of the Workshop of the UK Planning and Scheduling Special Interest Group, Nottingham, UK, (2006)
10. Little, I., Thiébaux, S.: Probabilistic planning vs replanning. In: Proceedings of the ICAPS'07 Workshop on Planning Competitions, Providence, RI, USA, (2007)
11. Teichteil-Königsbuch, F., Kuter, U., Infantes, G.: Incremental plan aggregation for generating policies in MDPs. In: Proceedings of the International Conference on Antonomous Agents and Multiagent Sytems, Toronto, Canada, (2010)
12. Yoon, S., Fern, A., Givan, R.: FF-replan: a baseline for probabilistic planning. In: Proceedings of the International Conference on Automated Planning and Scheduling, Providence, RI, USA, (2007)
13. Yoon, S., Fern, A., Givan, R., Kambhampati, S.: Probabilistic planning via determinization in hindsight. In: Proceedings of the AAAI Conference on Artificial Intelligence, Chicago, IL, USA, (2008)
14. Yoon, S., Ruml, W., Benton, J., Do, M.: Improving determinization in hindsight for on-line probabilistic planning. In: Proceedings of the International Conference on Automated Planning and Scheduling, Toronto, Ontario, Canada, (2010)
15. Younes, H.L.S., Littman, M.L., Weissman, D., Asmuth, J.: The first probabilistic track of the International Planning Competition. J. Artif. Intell. Res. **24**, 841–887 (2005)