The Markov Transition Constraint

Michael Morin and Claude-Guy Quimper

Department of Computer Science and Software Engineering
Université Laval, Québec, Qc, Canada
Michael.Morin.3@ulaval.ca
Claude-Guy.Quimper@ift.ulaval.ca

Abstract. We introduce a novel global Markov transition constraint (MTC) to model finite state homogeneous Markov chains. We present two algorithms to filter the variable domains representing the imprecise probability distributions over the state space of the chain. The first filtering algorithm is based on the fractional knapsack problem and the second filtering algorithm is based on linear programming. Both of our filtering algorithms compare favorably to the filtering performed by solvers when decomposing an MTC into arithmetic constraints. Cases where the fractional knapsack decomposition enforces bounds consistency are discussed whereas the linear programming filtering always perform bounds consistency. We use the MTC constraint to model and solve a problem of path planning under uncertainty.

1 Introduction

We introduce a novel global *markov transition constraint* (MTC) to model finite state space Markov processes with a finite number of steps T. Markov processes (also called Markov chains) are central to many applications. They are used in physics to model the motion of a target in a physical environment [22] and in computer science they are used in the Pagerank algorithm used by Google [20]. They are also used in economics and business science [8]. Markov processes even apply to arts for the generation of melodies [19] and lyrics [1]. They form the basis of decision making frameworks, such as Markov decision processes and hidden Markov decision processes which are fundamental to many artificial intelligence applications [16].

Let $\mathcal{N} = \{1, \ldots, N\}$ be a space of N states. States are mutually exclusive and jointly exhaustive, i.e., the process is in exactly one state of $\mathcal{N}$ at any time. Let $\mathbf{M}$ be the $N \times N$ *transition matrix* of the Markov process. That is, the probability of moving from state i to state j at any step $t \in \{1, \ldots, T\}$ is $0 \leq \mathbf{M}_{ij} \leq 1$ $(\forall i, j \in \mathcal{N})$, and the total probability of moving from state $i \in \mathcal{N}$ (given that the process is in state i) is 1, i.e., $\sum_{j \in \mathcal{N}} \mathbf{M}_{ij} = 1$ $(\forall i \in \mathcal{N})$. Let $\mathbf{x}^t = [x_1^t, \ldots, x_N^t]$ be the row vector of the probability distribution on the states at a time $t \in \{1, \ldots, T\}$ where x_i^t is the probability that the process is in state $i \in \mathcal{N}$ at step $t \in \{1, \ldots, T\}$. Given an initial distribution $\mathbf{x}^1$ over the states

such that $\sum_{i \in \mathcal{N}} x_i^1 = 1$ and $0 \leq x_i^1 \leq 1$ for all $i \in \mathcal{N}$, the *Markov property* states that

$$\mathbf{x}^{t+1} = \mathbf{x}^t \mathbf{M}, \qquad \forall t \in \{2, \ldots, T\}. \tag{1}$$

That is, the state at time $t+1$ depends on the previous state only.[1] Given the distribution $\mathbf{x}^t$, the distribution after k steps from step t, $\mathbf{x}^{t+k}$, is computed as:

$$\mathbf{x}^{t+k} = \mathbf{x}^t \mathbf{M}^k, \qquad \forall t \in \{1, \ldots, T\}, k \in \{0, \ldots, T-t, \}. \tag{2}$$

Example 1. Suppose that we wish to model the motion of a lost child between three stores (store 1, 2, and 3) of a mall with discrete time intervals of one minute. The child's behavior is as follows: s/he may spend time in the toys store (number 1); after one minute, her/his probability of leaving to the candy store (number 2) is $\frac{1}{8}$; from the candy store, s/he either returns to the toys store, stays there, or goes to the food market (number 3); whenever the child is in the food market, s/he directly returns to the candy store. The child's motion model is a transition matrix $\mathbf{M}$:

$$\mathbf{M} = \begin{bmatrix} \frac{7}{8} & \frac{1}{8} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & 0 \end{bmatrix}. \tag{3}$$

As a first experiment, suppose that the child starts in room 1. That is, the probability distribution at time 1 is $\mathbf{x}^1 = [1, 0, 0]$. We may, using equation (1), infer that the distribution over the child's location after a minute is $\mathbf{x}^2 = \left[\frac{7}{8}, \frac{1}{8}, 0\right]$. It is as easy to know, using equation (2), the distribution over the child's location after a delay of $k > 1$ minutes.

As a second experiment, suppose that we have no information on the child's initial location, but that we want to get the best possible estimation over her/his location after 1 minute, i.e., at time $t = 2$. The probability distribution at time 1 is uncertain. The only thing we know is that $\mathbf{x}^1$ is a distribution: it sums up to 1 and the probabilities are in the interval $[0, 1]$. By relying on linear optimization techniques or on the fractional knapsack filtering algorithm we present in Section 4, we find that, the probabilities of locating the child in room 1, 2 or 3 are in the intervals $\left[0, \frac{7}{8}\right]$, $\left[\frac{1}{8}, 1\right]$ and $\left[0, \frac{1}{3}\right]$ respectively; a solution that interval arithmetic alone is not able to provide. □

Clearly, filtering the uncertain probability distributions to the tightest possible intervals is not as easy in the second experiment as computing the probabilities in the first example. Uncertainty on probability distributions arises in modeling and problem solving due to external factors influencing the chain, imprecise data, and/or uncertain knowledge. For instance, in the search operation model presented in Section 6, the probability distributions are updated given

[1] Even if some generalizations allow the current state to depend on $d \geq 1$ states (e.g., Markov chains of order d), we restrict ourselves to previous state dependencies only, i.e., to first-order Markov chains.

the searcher's actions. There is no way, without knowing the entire searcher's path, to compute the exact probability distribution for each time step.

The introduction of a global constraint to model the evolution of the state space enables to intuitively model a Markov chain and leads to stronger filtering of the constrained variables when compared to a decomposition of the chain into elementary constraints.

The paper is organized as follows. We describe the MTC global constraint in Section 2. Section 3 discusses the literature related to similar constraints. In Section 4, we introduce two filtering algorithms: the first one based on linear optimization and the second one inspired from a fractional knapsack algorithm. We also discuss about the filtering of a decomposition into linear constraints. We identify cases where the different filtering techniques achieve bounds consistency. In Section 5, we conduct an empirical study of the filtering algorithms. Section 6 presents an application of the MTC constraint to a practical problem. We conclude in Section 7.

2 Modeling Markov transition processes as a global constraint

We define a new constraint that encodes a Markov transition process. As a convention, we write constrained variables in italic upper case and global constraints in small capitals. The MTC is defined as follows:

$$\textsc{Mtc}([Y_1,\ldots,Y_N],[X_1,\ldots,X_N],\mathbf{M}) \Leftrightarrow \forall j \in \mathcal{N}, \sum_{i\in\mathcal{N}} X_i\mathbf{M}_{ij} = Y_j \wedge \sum_{i\in\mathcal{N}} X_i = 1.$$

The matrix $\mathbf{M}$ is a known Markovian transition matrix. The vectors $\mathbf{X} = [X_1,\ldots,X_N]$ and $\mathbf{Y} = [Y_1,\ldots,Y_N]$ are probability distributions over $\mathcal{N}$. The domains of the variables in vectors $\mathbf{X}$ and $\mathbf{Y}$ are:

$$\text{dom}(X_i) = \left[\underline{X}_i, \overline{X}_i\right], \qquad \forall i \in \mathcal{N}, \tag{4}$$

where $\underline{X}_i$ and $\overline{X}_i$ are lower and upper bounds on variable X_i; and

$$\text{dom}(Y_j) = \left[\underline{Y}_j, \overline{Y}_j\right], \qquad \forall j \in \mathcal{N}, \tag{5}$$

where $\underline{Y}_j$ and $\overline{Y}_j$ are lower and upper bounds on variable Y_j.

The constraint $\textsc{Mtc}(\mathbf{Y},\mathbf{X},\mathbf{M})$ applies a transition matrix $\mathbf{M}$ to $\mathbf{X}$ and obtains $\mathbf{Y}$, i.e., $\textsc{Mtc}(\mathbf{Y},\mathbf{X},\mathbf{M})$ states that $\mathbf{Y} = \mathbf{X}\mathbf{M}$. Multiple MTC constraints may be chained to compute a finite Markov chain of T steps:

$$\textsc{Mtc}(\mathbf{X}^t,\mathbf{X}^{t-1},\mathbf{M}), \qquad \forall t \{2,\ldots,T\}. \tag{6}$$

Other constraints may be added on the X_i variables to interact with the chain.

Since the constraint $\textsc{Mtc}(\mathbf{Y},\mathbf{X},\mathbf{M})$ is satisfied only if $\mathbf{X}$ and $\mathbf{Y}$ represent probability distributions, filtering this constraint can both be seen as an application of the *theory of interval-probability* and as a linear optimization problem [12,24].

Definition 1 (Uncertain distribution). *We call a probability distribution an* uncertain distribution *if at least one of its probability is defined as an interval.*

For instance, the vector $[\mathrm{dom}(X_1), \ldots, \mathrm{dom}(X_n)]$ is a vector of intervals which is an uncertain distribution if and only if there exists at least an assignment to the variables X_i that sums up to 1.

3 Markov constraints and related literature

Pachet and Roy [18] introduced the *elementary markov constraint* (EMC). The EMC is defined as follows[2]:

$$\textsc{Emc}(S, S', P_{S'}) \Leftrightarrow P_{S'} = \mathbf{M}_{SS'}. \tag{7}$$

That is, $\textsc{Emc}(S, S', P_{S'})$ states that the probability of moving from state S to state S' is $P_{S'}$. The domains of the state variables S and S' are subsets of $\mathcal{N}$. The domain of the probability variable is the set of conditional probabilities, computed during the generation of the model, of achieving state S' from any previous state: $\mathrm{dom}(P_{S'}) = \left\{p \mid \left(\exists i \in \mathcal{N} \mid p = \mathbf{M}_{iS'}\right)\right\}$.

Using multiple EMCs, the authors model *constrained Markov processes*, i.e., Markov processes with supplementary constraints on the generated sequence. Let $S_1, \ldots, S_T$ be the state variables of a sequence of a first-order Markov chain. Let $P_{S_2}, \ldots, P_{S_T}$ be the variables that represent the probabilities. The constrained sequence is modeled as chained EMCs:

$$\textsc{Emc}(S_t, S_{t+1}, P_{S_{t+1}}), \qquad \forall t \in \{1, \ldots, T-1\}. \tag{8}$$

The Markov property, enforced by the EMCs, is a cost function to optimize whereas supplementary constraints are used to steer the generation of the sequence. The approach is used on chords sequence and melody generation.

Following [18], Pachet et al. [19] show that when the scope of the supplementary constraints does not exceed the chain's order d, the constrained Markov process may be recompiled into a statistically equivalent unconstrained Markov process. The approach is illustrated on the melody generation problem. In [1], the authors apply constrained Markov processes to the generation of lyrics.

The *markov transition constraint* (MTC) we introduce differs from the *elementary markov constraint* (EMC) presented in [18,19]. The former aims at modeling the evolution of the state space by computing the distributions $\mathbf{x}^t$ defined for all $t \in \{1, \ldots, T\}$ while the latter aims at generating the sequence of states in a constrained fashion, i.e., a Markov sequence with supplementary constraints. Moreover, the MTC deals with interval-domain probabilities while the EMC deals with finite-domain probabilities.

Markov chains with uncertain data (imprecise Markov chains) are related to Markov constraints. Imprecise Markov chains are provided a credal set (or

[2] We restraint our EMC definition to first-order Markov chains even though the definition found in [18] is general.

probability interval) for each possible transition whereas we deal with precise (singleton) transition probabilities and uncertain distributions. Further details on imprecise Markov chains may be found in [5,6].

4 Filtering the Markov Transition Constraint

A filtering algorithm for the MTC prunes the values from the domains of the variables X_i and Y_i that are inconsistent with the constraint.

Definition 2 (Interval support). *Let $C([X_1, \ldots, X_n])$ be a constraint of arity n. The assignment $[X_1, \ldots, X_n] = [x_1, \ldots x_n]$ is an interval support if and only if $C([x_1, \ldots x_n])$ is satisfied and the inequalities $\underline{X}_i \leq x_i \leq \overline{X}_i$ hold.*

Definition 3 (Bounds consistency). *A constraint $C([X_1, \ldots, X_n])$ is* bounds consistent *if and only if, for every variable X_i, there exists an interval support where the variable is assigned to the lower bound of its domain and a bounds support where the variable is assigned to the upper bound of its domain.*

A filtering algorithm *enforces bounds consistency* if and only if after being executed, no interval supports are eliminated and the constraint is bounds consistent. From now on, we simply say that an assignment has a *support* instead of an *interval support*.

We present three filtering algorithms to filter the variables X_i and Y_j ($\forall i, j \in \mathcal{N}$) subject to an MTC constraint. The first algorithm, denoted MTC-IA, uses the interval arithmetic [14] that is applied on a decomposition of the constraint. Following [24], the second algorithm, denoted MTC-LP, performs a linear optimization to achieve bounds consistency. The third algorithm, denoted MTC-FK, is a compromise between the two previous approaches. It relaxes the problem into a set of fractional knapsack constraints on which it enforces bounds consistency.

4.1 Filtering using interval arithmetic (MTC-IA)

We decompose the global constraint $\text{MTC}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$ into linear constraints as follows:

$$\sum_{i \in \mathcal{N}} X_i \mathbf{M}_{ij} = Y_j, \qquad \forall j \in \mathcal{N}, \quad (9)$$

$$\sum_{i \in \mathcal{N}} X_i = 1, \qquad (10)$$

$$\sum_{j \in \mathcal{N}} Y_j {\mathbf{M}^{-1}}_{ij} = X_i, \qquad \forall i \in \mathcal{N}, \quad (11)$$

$$\sum_{j \in \mathcal{N}} Y_j = 1. \qquad (12)$$

Constraints (9) and (10) follow from the definition of the MTC. Constraints (11) and (12) are implied constraints that enhance the filtering. We call MTC-IA

the algorithm that uses the interval arithmetic to enforce bounds consistency on the constraints (9) to (12). This algorithm, that is already implemented in most constraint solvers, simply enforces bounds consistency on each constraint until a fixed point is reached. The implied constraints necessitate the inverse of the transition matrix $\mathbf{M}$. The inverse $\mathbf{M}^{-1}$ is computed during the generation of the model, prior to solving the problem. This pre-solving process is done in cubic time.

We discuss specific cases where interval arithmetic enforces bounds consistency on the constraint Mtc. These cases require new definitions.

Definition 4 (Monomial matrix). *A matrix* $\mathbf{A}$ *is* monomial *if and only if it has one and only one non-null element in each column and each row.*

Proposition 1. *A monomial transition matrix* $\mathbf{M}$ *is a permutation matrix.*

Proof. The rows of $\mathbf{M}$ sums up to 1 by definition. Because $\mathbf{M}$ is monomial, we must have one element set to one in every row and all other elements are null which result into a binary matrix. Monomial binary matrices are permutation matrices. □

Proposition 2. *The inverse of a monomial transition matrix* $\mathbf{M}$ *is a transition matrix.*

Proof. $\mathbf{M}$ is a permutation matrix. The inverse of a permutation matrix is its transpose which is also a permutation matrix. □

Lemma 1. *If* $\mathbf{M}$ *is monomial, then enforcing bounds consistency on the linear constraints* (9) *and* (10) *enforces bounds consistency on* $\textsc{Mtc}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$.

Proof. If $\mathbf{M}$ is monomial, it is a permutation matrix and the constraints (9) are binary equalities. Suppose that constraints (9) and (10) are bounds consistent. Let x be an interval support for (10) then $y = x\mathbf{M}$ is a permutation of x and, thanks to the equality constraints (9), we have $y_i \in \text{dom}(Y_i)$. Consequently, the upper bound and lower bounds of the domains $\text{dom}(X_i)$ have an interval support for $\textsc{Mtc}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$. Let π be the permutation encoded by $\mathbf{M}$. Since the bounds of $\text{dom}(Y_{\pi(i)})$ are equal to the bounds of $\text{dom}(X_i)$, the bounds of $\text{dom}(Y_i)$ also have an interval support for $\textsc{Mtc}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$. □

Thanks to Proposition 2, Lemma 1 also holds when replacing constraint (9) by constraint (11) and/or constraint (10) by constraint (12).

4.2 Filtering using linear programming (MTC-LP)

In this section, we describe our linear programming (LP) reformulation of the filtering problem for the Mtc global constraint. Even though LP has not been applied (to our knowledge) to the filtering of global constraints encoding a Markov chain, there exist examples in the literature (e.g., [3] and [4]) where LP is used

to filter global constraints. These successes justify the application of the idea to Markov chains.

The LP filtering algorithm solves two linear programs per variable: one for the lower bound and one for the upper bound. To obtain a lower bound on variable X_i, the LP minimizes X_i subject to the constraints (14) to (17).

$$\min X_i \tag{13}$$

subject to

$$\sum_{i \in \mathcal{N}} \mathbf{M}_{ij} X_i = Y_j, \qquad \forall j \in \mathcal{N}, \tag{14}$$

$$\sum_{i \in \mathcal{N}} X_i = 1, \tag{15}$$

$$\underline{X}_i \leq X_i \leq \overline{X}_i, \qquad \forall i \in \mathcal{N}, \tag{16}$$

$$\underline{Y}_j \leq Y_j \leq \overline{Y}_j, \qquad \forall j \in \mathcal{N}. \tag{17}$$

New bounds on $\overline{X}_i$, $\underline{Y}_i$, and $\underline{Y}_i$ follow from a modification of the objective function. For each state $i \in \mathcal{N}$, we have the following LPs:

- $\overline{X}_i = \max X_i$ (resp. $\overline{Y}_i = \max Y_i$) subject to constraints (14) to (17);
- $\underline{X}_i = \min X_i$ (resp. $\underline{Y}_i = \min Y_i$) subject to constraints (14) to (17).

Each of the $4N$ linear programs may be solved using the simplex method [7]. We call this filtering technique based on linear optimization MTC-LP.

If an exact LP method is used (e.g., the simplex algorithm), we obtain optimal bounds on the domains of both X and Y.

Theorem 1. *MTC-LP enforces bounds consistency on* $\textsc{Mtc}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$.

Proof. The proof is a direct consequence of using an exact method for solving the linear programs. □

4.3 Filtering using the fractional knapsack (MTC-FK)

The last filtering algorithm we present, denoted MTC-FK, is based on the fractional knapsack problem and improves on the filtering done by the interval arithmetic. It is inspired from the global knapsack constraint [9].

We consider, for some $l \in \mathcal{N}$, the pair of constraints $\sum_{i \in \mathcal{N}} X_i = 1$ and $\sum_{i \in \mathcal{N}} \mathbf{M}_{il} X_i = Y_l$. To compute an upper bound on the variable Y_l, one greedily assigns the largest possible values to the variables X_i that are multiplied by the greatest weights $\mathbf{M}_{il}$ while making sure that the constraint (10) is satisfied. On the other hand, to compute a lower bound on the variable Y_l, one needs to assign the largest values to the variables multiplied by the smallest weights.

Algorithm 1 propagates the constraints (9) to (12) as well as the knapsack constraints $(\sum_{i \in \mathcal{N}} X_i \mathbf{M}_{ij} = Y_j, \sum_{i \in \mathcal{N}} X_i = 1)$ and $(\sum_{j \in \mathcal{N}} Y_j \mathbf{M}^{-1}{}_{ij} = X_i, \sum_{j \in \mathcal{N}} Y_j = 1)$ until it reaches a precision of ϵ. Algorithm 2 computes a lower bound on Y_l (or X_l). Reversing the order of the iterations in the for loop makes Algorithm 2 compute an upper bound on Y_l (or X_l).

Function `MTC-FK`$([X_1,\ldots,X_N],[Y_1,\ldots,Y_N],\mathbf{M},\mathbf{M}^{-1})$

Input: A vector of variables that represents the current uncertain distribution over $\mathcal{N}$: $[X_1,\ldots,X_N]$; a vector of variables that represents the resulting uncertain distribution: $[Y_1,\ldots,Y_N]$; a transition matrix and its inverse: $\mathbf{M}$ and $\mathbf{M}^{-1}$.

Output: The vectors of probability variables with filtered domain: $[X_1,\ldots,X_N]$, and $[Y_1,\ldots,Y_N]$.

repeat

  for $i \in \mathcal{N}$ **do** $x_i^{\text{old}} \leftarrow \overline{X}_i - \underline{X}_i$;

  for $i \in \mathcal{N}$ **do** $y_i^{\text{old}} \leftarrow \overline{Y}_i - \underline{Y}_i$;

  Enforce bounds consistency on constraints (9) to (12);

  foreach $l \in \mathcal{N}$ **do**

    $\underline{Y}_l \leftarrow$ `Fk-FilterLowerBound` $([X_1,\ldots,X_N],Y_l,[\mathbf{M}_{1l},\ldots,\mathbf{M}_{Nl}])$;

    $\overline{Y}_l \leftarrow$ `Fk-FilterUpperBound` $([X_1,\ldots,X_N],Y_l,[\mathbf{M}_{1l},\ldots,\mathbf{M}_{Nl}])$;

    $\underline{X}_l \leftarrow$ `Fk-FilterLowerBound` $([Y_1,\ldots,Y_N],X_l,[\mathbf{M}^{-1}_{1l},\ldots,\mathbf{M}^{-1}_{Nl}])$;

    $\overline{X}_l \leftarrow$ `Fk-FilterUpperBound` $([Y_1,\ldots,Y_N],X_l,[\mathbf{M}^{-1}_{1l},\ldots,\mathbf{M}^{-1}_{Nl}])$;

until $\sqrt{\sum_{i\in\mathcal{N}}(x_i^{\text{old}} - \overline{X}_i + \underline{X}_i)^2} + \sqrt{\sum_{i\in\mathcal{N}}(y_i^{\text{old}} - \overline{Y}_i + \underline{Y}_i)^2} \leq \epsilon$;

return $[X_1,\ldots,X_N]$, $[Y_1,\ldots,Y_N]$;

Algorithm 1: The MTC-FK filtering algorithm

Lemma 2. *Let $\overline{m}_j$ and $\underline{m}_j$ be the greatest and smallest value in column j of the matrix $\mathbf{M}$. If $\left[\underline{m}_j, \overline{m}_j\right] \subseteq \text{dom}\left(Y_j\right)\ \forall j \in \mathcal{N}$ then any distribution $x_1,\ldots,x_N$ (i.e., any assignment to the variables of vector $\mathbf{X}$ that sums to one) has a support in* $\textsc{Mtc}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$.

Proof. Let $\mathbf{M}^j$ be the j^{th} column of $\mathbf{M}$. Since the components of $\mathbf{X}$ sums to one and that none are negative, the scalar product of $\mathbf{X}$ and $\mathbf{M}^j$ is a convex combination of the elements in $\mathbf{M}^j$. Consequently, the result lies in the convex hull of $\mathbf{M}^j$ and it cannot be greater nor smaller than any element in $\mathbf{M}^j$. □

Lemma 3. *Let $\overline{m}_j$ and $\underline{m}_j$ be the greatest and smallest value in column j of the matrix $\mathbf{M}$. If $\left[\underline{m}_j, \overline{m}_j\right] \subseteq \text{dom}\left(Y_j\right)\ \forall j \in \mathcal{N}$ then MTC-FK enforces bounds consistency on* $\textsc{Mtc}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$.

Proof. The algorithm MTC-FK enforces bounds consistency on the constraint $\sum_{i\in\mathcal{N}} X_i = 1$ so that the lower bound and upper bounds of the domains of X_i can each form an assignment of the variables X_i that sums to one. From Lemma 2, any assignment that sums to 1 can be extended to a support of $\textsc{Mtc}(\mathbf{Y}, \mathbf{X}, \mathbf{M})$. We now need to prove that the variables Y_i are fully pruned. If a bound of $\text{dom}(Y_i)$ is modified, this bound was computed by the Algorithm 2 which constructed a valid support for the constraint. If a bound of $\text{dom}(Y_i)$ is not filtered, then either Algorithm 2 computed the same bound (in which case, it has a support) or it computed a larger one. However, the second case cannot occur since, as seen in Lemma 2, the scalar product of any distribution with a column of $\mathbf{M}$ leads to a value in $\left[\underline{m}_j, \overline{m}_j\right] \subseteq \text{dom}(Y_j)$. □

Function FK-FilterLowerBound($[U_1, \ldots, U_N]$,$\underline{V}_l$, $[t_1, \ldots, t_N]$)

Input: A vector of variables that represents an uncertain distribution over $\mathcal{N}$: $[U_1, \ldots, U_N]$; a lower bound on the variable that represents the probability that the process is in state l after applying transitions: $\underline{V}_l$; a vector of the transition probabilities to state l: $[t_1, \ldots, t_N]$.

Output: A new lower bound on state l probability: $\underline{V}_l$.

for $i \in \mathcal{N}$ **do** $u_i \leftarrow \underline{U}_i$;
$\lambda \leftarrow 1 - \sum_{l \in \mathcal{N}} \underline{U}_l$;
for $k \in \mathcal{N}$ *in non-decreasing order of* t_k **do**
  $\delta \leftarrow \min\left(\lambda, \overline{U}_k - \underline{U}_k\right)$;
  $u_k \leftarrow u_k + \delta$;
  $\lambda \leftarrow \lambda - \delta$;
  if $\lambda = 0$ **then** break;
return $\max\left(\sum_{i \in \mathcal{N}} u_i t_i, \underline{V}_l\right)$;

Algorithm 2: The FK-FilterLowerBound lower bound filtering algorithm

Lemma 3 is particularly useful at the beginning of the search in a problem where the domains of the variables Y_i are the intervals $[0, 1]$.

Theorem 2. *The consistency achieved by each algorithm satisfies MTC-IA* $\prec$ *MTC-FK* $\prec$ *MTC-LP.*

Proof. We first prove MTC-IA $\preceq$ MTC-FK $\preceq$ MTC-LP. The MTC-FK algorithm filters the same constraints as MTC-IA but the knapsack constraints consider pairs of constraints which offer a filtering that is not weaker. The algorithm MTC-LP achieves bounds consistency which is optimal. We now show two examples which prove that MTC-IA $\neq$ MTC-FK, and that MTC-FK $\neq$ MTC-LP. Let $N = 3$. Let $\mathrm{dom}\,(X_1) = [.3, 1]$, and $\mathrm{dom}\,(X_2) = \mathrm{dom}\,(X_3) = [0, 1]$. Let $\mathrm{dom}\,(Y_1) = \mathrm{dom}\,(Y_2) = \mathrm{dom}\,(Y_3) = [0, 1]$. Let the transition matrix be

$$\mathbf{M} = \begin{bmatrix} 0 & .4 & .6 \\ .3 & .4 & .3 \\ .4 & .6 & 0 \end{bmatrix}. \tag{18}$$

By Lemma 2, MTC-FK enforces bounds consistency whereas MTC-IA does not. In fact, MTC-FK sets $\overline{Y}_1 = 0.28$ whereas MTC-IA sets $\overline{Y}_1 = 0.49$. Suppose that $\mathrm{dom}\,(Y_1) = [.1, 1]$. By Theorem 1, MTC-LP enforces bounds consistency which is not the case of MTC-FK. In fact, MTC-LP sets $\overline{X}_1 = 0.75$ while MTC-FK sets $\overline{X}_1 = 0.9$. □

5 Empirical experiments and discussion

We generated random domains and transition matrices to compare the three filtering algorithms and see how much their filtering differ. For MTC-IA, we

present the results with and without the implied constraints (11) and (12). The MTC-LP enforces bounds consistency on all instances providing the optimal bounds. Let $\underline{\mathbf{x}}^{\text{old}}$, $\overline{\mathbf{x}}^{\text{old}}$, $\underline{\mathbf{y}}^{\text{old}}$ and $\overline{\mathbf{y}}^{\text{old}}$ be the initial bounds of the filtering problem. Let $\underline{\mathbf{x}}^*$, $\overline{\mathbf{x}}^*$, $\underline{\mathbf{y}}^*$ and $\overline{\mathbf{y}}^*$ be the optimal bounds found by MTC-LP. Let $\underline{\mathbf{x}}$, $\overline{\mathbf{x}}$, $\underline{\mathbf{y}}$ and $\overline{\mathbf{y}}$ be the bounds found by a given filtering method. We define the *proportion of optimality* as the ratio of the sum of the distances traveled, in the domain space, by a filtering method to the sum of the distances traveled by MTC-LP:

$$Ind_{\text{p}} = \frac{\left\|\underline{\mathbf{x}} - \underline{\mathbf{x}}^{\text{old}}\right\| + \left\|\overline{\mathbf{x}}^{\text{old}} - \overline{\mathbf{x}}\right\| + \left\|\underline{\mathbf{y}} - \underline{\mathbf{y}}^{\text{old}}\right\| + \left\|\overline{\mathbf{y}}^{\text{old}} - \overline{\mathbf{y}}\right\|}{\left\|\underline{\mathbf{x}}^* - \underline{\mathbf{x}}^{\text{old}}\right\| + \left\|\overline{\mathbf{x}}^{\text{old}} - \overline{\mathbf{x}}^*\right\| + \left\|\underline{\mathbf{y}}^* - \underline{\mathbf{y}}^{\text{old}}\right\| + \left\|\overline{\mathbf{y}}^{\text{old}} - \overline{\mathbf{y}}^*\right\|}. \tag{19}$$

We generated five sets of filtering problem instances: a transition matrix $\mathbf{M}$ along with random bounds on the variables of vectors $\mathbf{X}$ and $\mathbf{Y}$. The randomly generated bounds are feasible, i.e., the constraint Mtc is satisfiable. Table 1 summarizes the sets' characteristics. The first set (*random*) contains randomly generated transition matrices. The second and the third sets (*star* and *plus grids*) contain transition matrices that represents square grids, i.e., states are located on a square grid and are only connected to their neighbors. *Plus grids* are grids where a state is linked to its North, South, West and East neighbors. *Star grids* also include diagonals (NW, NE, SW, SE). Let ρ be the conditional probability that the process stays in state i when it is in state i for any $i \in \mathcal{N}$, i.e., the *probability of stationarity* $\rho = \mathbf{M}_{ii}$ $(\forall i \in \mathcal{N})$. Let $\deg(i)$ be the degree of cell i (loops included) in the adjacency matrix of the grid. The transition matrix of a grid instance is defined as:

$$\mathbf{M}_{ij} = \begin{cases} \frac{1-\rho}{\deg(i)-1} & \text{if } i \neq j; \\ \rho & \text{if } i = j. \end{cases} \tag{20}$$

We chose $\rho \in \{.2, .4, .6, .8\}$. The fourth set (*zero-one-Y*) contains randomly generated transition matrices, but the domains of the variables of vector $\mathbf{Y}$ are $[0, 1]$. That is, the information about the future (i.e., the uncertain distribution of vector $\mathbf{Y}$) comes from the present (i.e., the uncertain distribution of vector $\mathbf{X}$). This is the usual way to process Markov chains in Markov processes. The fifth set (*zero-one-X*) models the converse case where the information about the present process' state comes from the future. Our benchmark library includes non-singular matrices only. We generated 10 different instances for each pair of state space size N and probability of stationarity ρ in each set for a total of 3690 instances.

Figures 1 and 2 show scatter plots comparing the proportion of optimality achieved by different filtering algorithms. The higher the proportion of optimality is on a given axis, the better the filtering algorithm performs (a value of 1 represents a bounds consistent domain as obtained by the MTC-LP algorithm). The dotted line is a visual frontier between the two compared algorithms' performance. A dot representing an instance for which both algorithms produce the same filtering lies on this visual frontier. A dot for which the algorithm on the

Table 1: The characteristics of the instance sets

Name	N	$\mathbf{M}$	$\rho = \mathbf{M}_{ii}$	Set size
Random	$\{2, \ldots, 100\}$	Random	Random	990
Star grids	$\{4, 9 \ldots, 100\}$	Star grids	$\{.2, .4, .6, .8\}$	360
Plus grids	$\{4, 9, \ldots, 100\}$	Plus grids	$\{.2, .4, .6, .8\}$	360
Zero-one-Y	$\{2, \ldots, 100\}$	Random	Random	990
Zero-one-X	$\{2, \ldots, 100\}$	Random	Random	990

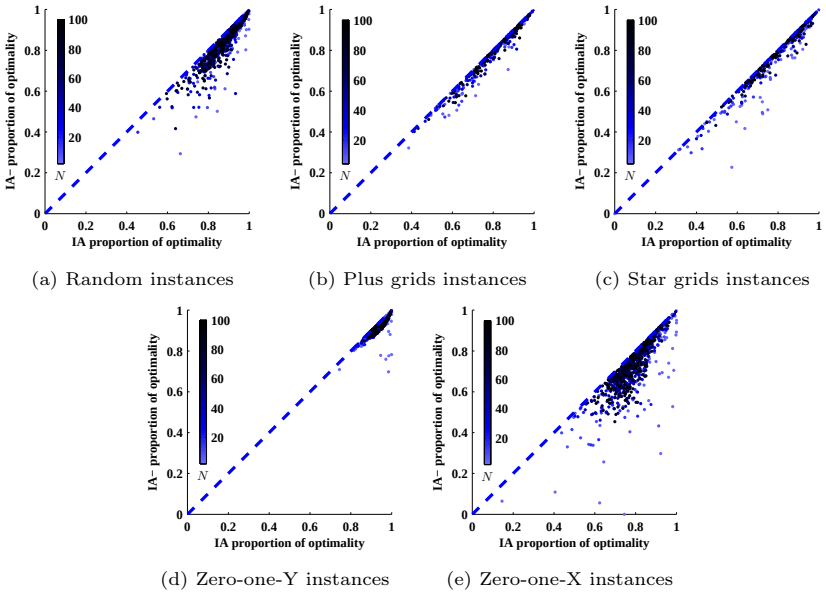
x-axis (y-axis) performs better than the algorithm on the y-axis (x-axis) lies on the right (left) hand-side of the frontier. The algorithm with the highest density of dots on its side of the frontier tends to achieve the best overall performance. Darker blue shades are used for instances with a larger state space size (N); lighter blue shades are used for instances with a smaller N.

As shown on Figure 1, MTC-IA (x-axis) outperforms MTC-IA without implied constraints (y-axis) on all instance sets. While the performance of the two algorithms is similar on some random instances (Figure 1a), the importance of implied constraints is clear as all the dots fall on the right hand side of the frontier. The scatter plots of grid instance sets (Figures 1b and 1c) favor MTC-IA. Zero-one-Y instances represent forward in time inferences using $\mathbf{M}$ (Figure 1d). Zero-one-X instances represent backward in time inferences using $\mathbf{M}^{-1}$ (Figure 1e). On these instances, we see the benefits of the interaction between a set of elementary constraints: the implied constraints enable the algorithm to further filter the domains backward in time whenever knowledge on the future is acquired by forward filtering. The difficulty of backward inferences is shown by the fact that the distribution of the results on the Zero-one-Y instances (forward inference) is closer to 1 when compared to the distribution of the results on the Zero-one-X instances (backward inference). This is partly due to the negative values in the inverse of most transition matrix $\mathbf{M}$.

As shown on Figure 2, MTC-FK (x-axis) outperforms MTC-IA (y-axis) on all instance sets. The performance of both algorithms is close on the Random set (Figure 2a), but still, MTC-FK performs better. The same tendency appears for grids (Figures 2b and 2c). MTC-FK enforced bounds consistency on all Zero-one-Y instances (a result of Lemma 3) whereas this is not the case for MTC-IA. Forward inference is easier than backward inference for both algorithms: the distribution of the optimality results is closer to 1 on the Zero-one-Y set than on the Zero-one-X set.

6 An application to path planning under uncertainty

We illustrate the Mtc constraint usage on an *optimal search path* (OSP) problem [15]. A searcher moves on a graph $G_A = (\mathcal{V}(G_A), \mathcal{E}(G_A))$, within T time steps, in order to find a lost object. In absence of search, the object would simply



(a) Random instances (b) Plus grids instances (c) Star grids instances

(d) Zero-one-Y instances (e) Zero-one-X instances

Fig. 1: Proportion of optimality achieved by MTC-IA (IA) when compared to MTC-IA without implied constraints (IA-)

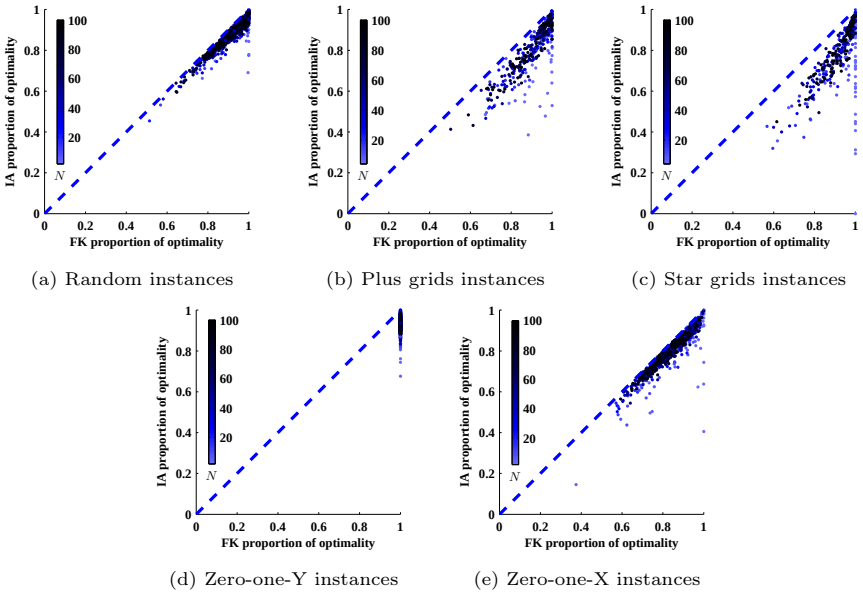
move from vertex to vertex according to a transition matrix $\mathbf{M}$. The searcher, however, influences the chain evolution by searching the vertices and by removing the object as soon as it detects it: a special "removed" state that represents the searcher's hands is added to the state space.

The probability that a vertex r contains the object at a time t is called the *probability of containment*, or $poc_t(r)$. The initial distribution (poc_1) is known a priori. The distribution $\mathbf{x}_t$ over the states is

$$\mathbf{x}_t = [poc_t(1), \ldots, poc_t(N), cos_{t-1}]\,, \tag{21}$$

where cos_t is the searcher's *cumulative overall probability of success* linked to the removed state at time t ($cos_0 = 0$). When the searcher is located in a given vertex r at time t, i.e., $y_t = r$, s/he searches that vertex. Her/his known probability of detection (pod) is conditional to the object's presence. The local success of the searcher in r at time t is defined as:

$$pos_t(r) = \begin{cases} poc_t(r) \times pod & \text{if } r = y_t, \\ 0 & \text{if } r \neq y_t. \end{cases} \tag{22}$$



(a) Random instances (b) Plus grids instances (c) Star grids instances

(d) Zero-one-Y instances (e) Zero-one-X instances

Fig. 2: Proportion of optimality achieved by MTC-FK (FK) when compared to MTC-IA (IA)

We split the chain to introduce a distribution $\widehat{\mathbf{x}}^t$ that represents the search and to apply the object's motion:

$$\widehat{\mathbf{x}}^t = [poc_t(1) - pos_t(1), \ldots, poc_t(N) - pos_t(N), cos_t], \tag{23}$$

$$\mathbf{x}^{t+1} = \widehat{\mathbf{x}}^t \begin{bmatrix} \mathbf{M} & 0 \\ 0 & 1 \end{bmatrix}. \tag{24}$$

The searcher's goal is to maximize cos_T. The OSP problem is known to be $\mathcal{NP}$-hard even for stationary objects [23]. Recent examples of OSP-related researches with a single searcher may be found in the works of [2,11,21].

This leads to a CP model with four sets of interval-domain probability variables: the variables COS_t that represent the probability of finding the object up to time t, and the variables $POC_t(r)$, $POC_t^{\text{Searched}}(r)$, and $POS_t(r)$ that represent the probabilities in the split Markov chain ($\forall t \in \{1, \ldots, T\}, r \in \mathcal{V}(G_A)$). A set of finite-domain variables $PATH_t$ models the searcher's path. The domains of each path variable is a subset of vertices, i.e., $\text{dom}(PATH_t) \subseteq \mathcal{V}(G_A)$ ($\forall t \in \{1, \ldots, T\}$). $POC_1(r) = poc_1(r)$, and $PATH_0 = y_0$ are known data. The probability of finding (removing) the object up to time t is:

$$COS_t = \sum_{1 \leq t' \leq t} \max_{r \in \mathcal{V}(G_A)} POS_{t'}(r). \tag{25}$$

The objective is to maximize COS_T subject to the graph edges constraints:

$$(PATH_{t-1}, PATH_t) \in \mathcal{E}(G_A), \qquad \forall t \in \{1, \ldots, T\}; \quad (26)$$

the probability of success along the path of the searcher:

$$PATH_t = r \implies POS_t(r) = POC_t(r)pod, \qquad \forall t \in \{1, \ldots, T\}, \forall r \in \mathcal{V}(G_A); \quad (27)$$

$$PATH_t \neq r \implies POS_t(r) = 0, \qquad \forall t \in \{1, \ldots, T\}, \forall r \in \mathcal{V}(G_A); \quad (28)$$

the effect of searching on the chain:

$$POC_t^{\text{Searched}}(r) = POC_t(r) - POS_t(r), \qquad \forall t \in \{1, \ldots, T\}, \forall r \in \mathcal{V}(G_A); \quad (29)$$

and the application of the Markovian transition matrix of the object on the split chain:

$$\text{MTC}\left(\mathbf{X}^{t+1}, \widehat{\mathbf{X}}^t, \begin{bmatrix} \mathbf{M} & 0 \\ 0 & 1 \end{bmatrix}\right), \qquad \forall t \in \{1, \ldots, T-1\}, \forall r \in \mathcal{V}(G_A), \quad (30)$$

where

$$\mathbf{X}^{t+1} = [POC_{t+1}(1), \ldots, POC_{t+1}(N), COS_t], \quad (31)$$

and

$$\widehat{\mathbf{X}}^t = \left[POC_t^{\text{Searched}}(1), \ldots, POC_t^{\text{Searched}}(N), COS_t\right]. \quad (32)$$

6.1 Applied experiment and discussion

We implemented two Markov chain-based models for the OSP using Choco Solver 2.1.3 [10], a state-of-the-art CP solver. The first, OSP-IA-, uses a standard elementary arithmetic constraints decomposition in (30). The second, OSP-FK, uses fractional knapsack filtering in (30). We kept the model as close as possible to the one presented in [15] while adding the necessary variables to model the search with MTCs. The goal of the experiment is to show the benefits of the filtering achieved by the MTC when compared to elementary arithmetic constraint decomposition. All implementations of the application are done using the Java programming language, the Apache Commons Math [13] library, and the Java Universal Network/Graph (JUNG) 2.0.1 framework [17]. The total detection value selection heuristic developed for the OSP problem [15] is used.

The accessibility graphs (G_A) are the following: G^+, a 11×11 plus grid, G^* a 11×11 star grid, and the University Laval tunnels map G^L [15]. We allowed a total of $T = 17$ time steps. The probability of detection are $pod(r) \in \{0.3, 0.6, 0.9\}$ ($\forall r \in \mathcal{V}(G_A)$). The assumed Markovian object's motion model of the objective, i.e., its transition matrix $\mathbf{M}$, is based on equation (20). The probability of stationary of the object in $\mathbf{M}$ are $\rho \in \{0.3, 0.6, 0.9\}$. We allowed a

total number of 5,000,000 backtracks and a time limit of 20 minutes. We tested with different epsilon values and retained $\epsilon = 0.3$ as it is sufficient in practice for most search operations. All tests were run on an Intel(R) Core(TM) i7-2600 CPU with 4 GB of RAM.

Table 2 compares standard filtering (OSP-IA-) to the fractional knapsack filtering (OSP-FK). We chose three performance criteria: the objective value (COS), the time consumed to the last incumbent solution, and the total number of backtracks to obtain the last incumbent solution. The objective value and the time to last incumbent tell us which model achieves the highest performance on a given problem instance independently of its filtering performance. The total number of backtracks to last incumbent and the time consumed tell us whether the model filtering performance is good or not. For each criterion, underline results belongs to the best performing techniques. We do not underline ties. Overall, OSP-FK outperformed OSP-IA- by achieving a higher or equal objective values while maintaining a lower number of backtracks on most instances. Even though we demonstrated that the theoretical consistency achieved by MTC-FK is better than the one achieved by MTC-IA (with and without implied constraints), some cases may occurs where MTC-IA (with or without implied constraints) outperforms MTC-FK on applications. This may be due to the interaction between a heuristic (e.g., the total detection heuristic) and the solving process. Efficient heuristics lead to smaller search trees removing some filtering needs. As demonstrated by our positive results, thorough filtering of the variables is beneficial in most cases. The MTC-FK filtering algorithm achieves high quality solutions soon in the solving process while greatly reducing the total number of backtracks when compared to a model with elementary constraints.

7 Conclusion

We introduced Mtc, a global constraint that models Markov chains. We discussed cases where elementary arithmetic constraints enforce bounds consistency. We proved that elementary arithmetic constraints, even with implied constraints, are insufficient to enforce bounds consistency in all cases and discussed a linear programming algorithm that always performs bounds consistency. However, a constraint solver needs to run the filtering algorithms on an exponential number of nodes, and linear optimization turns out to be an expensive filtering solution. Thus, we provided, as a trade-off, a filtering algorithm based on the fractional knapsack problem. The method is proved to achieve bounds consistency when the transition matrix is monomial and in the case of forward reasoning. Finally, we successfully applied the Mtc global constraint on the OSP problem, a path planning problem under uncertainty involving Markov transitions.

Table 2: Results on OSP problem instances with $\epsilon = 10^{-3}$; underlined values are better.

$pod(r)$	ρ	**Objective** (COS)		**Time**† **(s)**		**Backtracks**	
		IA-	FK	IA-	FK	IA-	FK
G^L *with* $T = 17$							
0.3	0.3	0.519	0.519	1	1	1948	1
	0.6	0.578	0.578	1	2	1880	2
	0.9	0.738	0.738	1	1	1550	11
0.6	0.3	0.735	0.735	1	1	1980	10
	0.6	0.758	0.758	1	1	1851	9
	0.9	0.845	0.845	1	1	1475	35
0.9	0.3	0.810	0.810	1	1	1911	23
	0.6	0.835	0.835	1	1	1832	35
	0.9	0.890	0.890	1	1	1396	61
G^+ *with* $T = 17$							
0.3	0.3	0.106	0.106	4	7	7205	1
	0.6	0.165	0.165	3	5	5343	0
	0.9	0.442	0.442	3	5	2236	20
0.6	0.3	0.189	0.189	4	7	7044	2
	0.6	0.298	0.298	3	5	5241	0
	0.9	0.551	0.551	3	5	2325	43
0.9	0.3	0.259	0.259	4	7	6946	2
	0.6	0.329	0.329	3	5	5130	0
	0.9	0.623	0.623	3	5	2152	73
G^* *with* $T = 17$							
0.3	0.3	0.131	0.131	5	9	7757	1
	0.6	0.219	0.219	4	7	6444	0
	0.9	0.584	0.584	3	6	3324	35
0.6	0.3	0.226	0.226	5	10	7514	6
	0.6	0.314	0.348	4	7	6334	0
	0.9	0.686	0.686	3	6	3344	58
0.9	0.3	0.304	0.301	5	9	7209	9
	0.6	0.381	0.381	4	7	6065	2
	0.9	0.734	0.734	3	6	2945	54

† The time to last incumbent solution.

References

1. Barbieri, G., Pachet, F., Roy, P., Degli Esposti, M.: Markov constraints for generating lyrics with style. In: Proceedings of 20th biennial European Conference on Artificial Intelligence, IOS Press (2012)
2. Berger, J., Lo, N., Noel, M.: Exact solution for search-and-rescue path planning. International Journal of Computer and Communication Engineering **2**(3) (may 2013)
3. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: Filtering algorithms for the NValue constraint. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer (2005) 79–93
4. Bessiére, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: Filtering algorithms for the NValue constraint. Constraints **11**(4) (2006) 271–293

5. Blane, H., Den Hertog, D.: On markov chains with uncertain data. Available at SSRN 1138144 (2008)
6. De Cooman, G., Hermans, F., Quaeghebeur, E.: Imprecise markov chains and their limit behavior. Probability in the Engineering and Informational Sciences **23**(4) (2009) 597
7. Garfinkel, R.S., Nemhauser, G.L.: Integer programming. Volume 4. Wiley New York (1972)
8. Hamilton, J.: A new approach to the economic analysis of nonstationary time series and the business cycle. Econometrica: Journal of the Econometric Society (1989) 357–384
9. Katriel, I., Sellmann, M., Upfal, E., Van Hentenryck, P.: Propagating knapsack constraints in sublinear time. In: Proceedings of the national conference on artificial intelligence (AAAI-07). (2007) 231–236
10. Laburthe, F., Jussien, N.: Choco Solver Documentation. (2012) `http://www.emn.fr/z-info/choco-solver/`.
11. Lau, H., Huang, S., Dissanayake, G.: Discounted mean bound for the optimal searcher path problem with non-uniform travel times. European journal of operational research **190**(2) (2008) 383–397
12. Luis, M., Campos, D., Huete, J., Moral, S.: Probability intervals: a tool for uncertain reasoning. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **2**(02) (1994) 167–196
13. Apache Commons: Commons math: The apache commons mathematics library (2010) Accessed: 2013-10.
14. Moore, R.E.: Interval analysis. Volume 2. Prentice-Hall Englewood Cliffs (1966)
15. Morin, M., Papillon, A.P., Laviolette, F., Abi-Zeid, I., Quimper, C.G.: Constraint programming for path planning with uncertainty: Solving the optimal search path problem. In Milano, M., ed.: Principles and Practice of Constraint Programming, Springer Berlin Heidelberg (2012) 988–1003
16. Norvig, P., Russell, S.J.: Artificial Intelligence: A Modern Approach. 3e edn. Prentice Hall (2013)
17. O'Madadhain, J., Fisher, D., Nelson, T., White, S., Boey, Y.: Jung: Java universal network/graph framework. `http://jung.sourceforge.net` (2010)
18. Pachet, F., Roy, P.: Markov constraints: steerable generation of markov sequences. Constraints **16**(2) (2011) 148–172
19. Pachet, F., Roy, P., Barbieri, G.: Finite-length markov processes with constraints. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One, AAAI Press (2011) 635–642
20. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Technical report, Stanford InfoLab (1999)
21. Sato, H., Royset, J.: Path optimization for the resource-constrained searcher. Naval Research Logitics **57**(5) (2010) 422–440
22. Stone, L.: Theory of Optimal Search. Academic Press, New York (2004)
23. Trummel, K., Weisinger, J.: The complexity of the optimal searcher path problem. Operations Research **34**(2) (1986) 324–327
24. Weichselberger, K.: The theory of interval-probability as a unifying concept for uncertainty. International Journal of Approximate Reasoning **24**(2) (2000) 149–170