Constraint Programming for Path Planning with Uncertainty

Solving the Optimal Search Path problem

Michael Morin[1], Anika-Pascale Papillon[2], Irène Abi-Zeid[3], François Laviolette[1], and Claude-Guy Quimper[1]

[1] Department of Computer Science and Software Engineering
[2] Department of Mathematics and Statistics
[3] Department of Operations and Decision Systems
Université Laval, Québec, Qc, Canada
{Michael.Morin.3, Anika-Pascale.Papillon.1} @ulaval.ca
Irene.Abi-Zeid@osd.ulaval.ca
{Francois.Laviolette, Claude-Guy.Quimper} @ift.ulaval.ca

Abstract. The optimal search path (OSP) problem is a single-sided detection search problem where the location and the detectability of a moving object are uncertain. A solution to this $\mathcal{NP}$-hard problem is a path on a graph that maximizes the probability of finding an object that moves according to a known motion model. We developed constraint programming models to solve this probabilistic path planning problem for a single indivisible searcher. These models include a simple but powerful branching heuristic as well as strong filtering constraints. We present our experimentation and compare our results with existing results in the literature. The OSP problem is particularly interesting in that it generalizes to various probabilistic search problems such as intruder detection, malicious code identification, search and rescue, and surveillance.

1 Introduction

The *optimal search path* (OSP) problem we address in this paper is a single-sided detection search problem where the location and the detectability of a moving search object are uncertain. The single-sided search assumption means that the object's movements are independent of the searcher's actions. In other words, the object does not act, neither to meet nor to escape the searcher. A solution to this $\mathcal{NP}$-hard problem [1] is a path on a graph that maximizes the probability of finding an object that moves according to a known motion model. In the OSP problem, a moving agent must plan its optimal path in order to detect a mobile search object subject to constraints. This is a path planning problem for a detection search with uncertainty on the whereabouts of the search object, on the detection capabilities of the searcher, and on the movement of the search object. This type of problem arises in many applications related to detection searches namely, search and rescue [2], military surveillance, malicious code detection [3],

covert messages (violating the security policies of the system) on the Internet [4], and locating a mobile user in a cellular network for optimal paging [5]. In this paper, we introduce *constraint programming* (CP) models that we developed in order to solve the OSP problem. We assume a single indivisible searcher where search effort corresponds to the time available for searching and a probability of detection is associated with each time step. Furthermore, the movement of the searcher is constrained to an accessibility graph.

Most work on the single searcher OSP problem in discrete time and space involved *branch and bound* (BB) algorithms. In [6], Stewart proposed a *depth-first* BB algorithm using a bound that does not guarantee optimality. Eagle [7] considered a Markovian object's motion model and proposed a dynamic programming approach. Eagle and Yee [8] presented an optimal bound for Stewart's BB algorithm. With an object following a Markovian motion model and an exponential *probability of detection* (*pod*) function, their approach produced an optimal bound by relaxing the search effort indivisibility constraint on a set of vertices while maintaining the path constraints. The bound is computed in polynomial time. A review of the BB algorithm procedures and of the OSP problem bounding techniques before 1998 can be found in [9]. Among the recent developments linked to the OSP problem, Lau *et al.* [10] proposed the DMEAN bound which was derived from the MEAN bound found by Martins [11].

The advantage of using CP in the OSP problem context lies in the CP model's expressivity. The model stays close to the formulation of the problem while enabling the use of problem specific constraints, heuristics and bounds. Furthermore, previous results on similar problems (*e.g.*, [12]) show that CP allows to find high quality solutions quickly, an interesting property we explore in this paper.

According to [13], uncertainty in constraint problems may arise in two situations:

- the problem changes over time (*dynamically changing* problems), and
- some problem's data or information are missing or are unclear (*uncertain* problems).

The OSP problem formulation as a constraint program is not uncertain in this sense since it has a complete description. Nonetheless, the location of the search object, its detectability, and its motion are represented by probability distributions. Our CP is not a dynamic formulation since the searcher's detection model, the object's motion model and the prior probability distribution on its location are known *a priori*. More specifically, the OSP problem is a path planning problem with a Markov Decision Process formulation that uses negative information for updating the probabilities in the absence of detection. In the case where the total number of plausible search object's paths is sufficiently low, a situation that rarely occurs in realistic search problems, the problem could potentially be formulated using multiple scenarios and thus be considered a stochastic CP (*e.g.*, [14]) where a scenario would correspond to a possible path of the search object. However, this is not an interesting approach since the Markov OSP problem specialization from search theory enables us to solve the problem without

enumerating all the object's plausible paths [15]. Surveys on dealing with uncertainty in constraint problems may be found in [13,16].

Section 2 presents the OSP formalism. Sections 3 and 4 respectively describe the proposed constraint program and the experimentation. The results are discussed in Section 5 and compared to existing results in the literature. We conclude in Section 6.

2 The OSP problem in its general discrete form

When solving the OSP problem, the goal is to find a path (a search plan), constrained by time, that maximizes the probability of detecting a moving object of unknown location. A continuous search environment may be discretized by a graph[4] $G_A = (\mathcal{V}(G_A), \mathcal{E}(G_A))$ where $\mathcal{V}(G_A)$ is a set of discrete regions. A vertex r is accessible from vertex s if and only if the edge (s, r) belongs to the accessibility graph G_A. The search operation is defined over a given finite set $\mathcal{T} = \{1, \ldots, T\}$ of time steps. Let $y_t \in \mathcal{V}(G_A)$ be the searcher's location at time $t \in \mathcal{T}$. When $y_t = r$, we say that vertex r is searched at time t with an associated probability of detection. A search plan P (*i.e.*, the sequence of vertices searched) is determined by the searcher's path on G_A starting at location $y_0 \in \mathcal{V}(G_A)$:

$$P = [y_0, y_1, \ldots, y_T]. \tag{1}$$

The unknown object's location is characterized by a probability of containment (*poc*) distribution over $\mathcal{V}(G_A)$ that evolves in time, due to the search object's motion and to updates following unsuccessful searches. The poc_1 distribution over $\mathcal{V}(G_A)$ is the *a priori* knowledge on the object's location. A local probability of success (*pos*) is associated with the searcher being located in vertex r at time t. It is the probability of detecting the object in vertex r at time t defined as:

$$pos_t(r) = poc_t(r) \times pod(r), \tag{2}$$

where $pod(r)$ is the probability, conditional to the object's presence in r at time t, of detecting the object in vertex r at time t. This detection model is known *a priori*. For all $t \in \mathcal{T}$, $r \in \mathcal{V}(G_A)$, the detection model is

$$pod(r) \in (0, 1], \qquad \text{if } y_t = r; \tag{3}$$
$$pod(r) = 0, \qquad \text{otherwise.} \tag{4}$$

The OSP formalism assumes that a positive detection of the object stops the search. The probabilities of containment change in time following an assumed

[4] We restrict ourselves to the case of undirected reflexive graphs (*i.e.*, every vertex has a loop) since they are more natural in search problems. Furthermore, loops enable the searcher and the object to stay at their current location instead of moving on.

Markovian object motion model $\mathbf{M}$ and according to the negative information collected on the object's presence. Thus, for all time $t \in \{2, \ldots, T\}$, we have that

$$poc_t(r) = \sum_{s \in \mathcal{V}(G_A)} \mathbf{M}(s, r) \left[poc_{t-1}(s) - pos_{t-1}(s) \right], \tag{5}$$

where $\mathbf{M}(s, r)$ is the probability of the object moving from vertex s to vertex r within one time step. The optimality criterion for a search plan P is the maximization of the global and cumulative success probability of the operation (COS) over all vertices and time steps defined as:

$$COS(P) = \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{V}(G_A)} pos_t(r). \tag{6}$$

2.1 An optimal search plan example

Figure 1 shows an example of an environment with doors and stairs accessibility. Considering the accessibility graph, and assuming $T = 5$, $y_0 = 3$, $poc_1(4) = 1.0$, $pod(y_t) = 0.9$ ($\forall t \in \mathcal{T}$) and a uniform Markovian motion model between accessible vertices, an optimal search plan P^* would then be

$$P^* = [y_0, y_1, \ldots, y_5] = [3, 6, 7, 7, 7, 7]. \tag{7}$$

Using Table 1, we explain why search plan P^* is optimal. Starting from vertex 3,

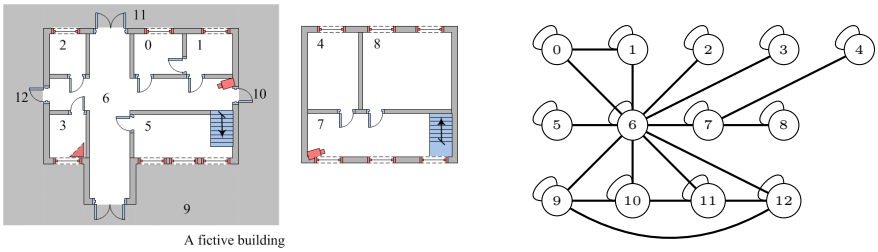

Fig. 1: A fictive building OSP problem environment (left) and its accessibility graph (right).

the searcher first moves to vertex 6 since the probability of containment is high in vertex 4. Then, the only accessible vertex where the probability of containment is nonzero is vertex 7. Therefore, the searcher moves from vertex 6 to vertex 7. Finally, the search plan stabilizes in vertex 7 since it has the highest probability of containment at each subsequent time step. The objective (COS) value of the optimal search plan P^* is equal to 0.889. For a search plan P, the objective value is computed as follows:

– compute the local probability of success in vertex y_1 at time step 1 ($pos_t(y_1)$) using Equation (2);
– for all vertices r, compute the probability of containment at time step 2 ($poc_2(r)$) using Equation (5);
– apply the same process for time steps 2 to T;
– sum all the local success probabilities obtained in time steps 1 to T to compute the objective (COS) value of the search plan P.

The objective value at time step t, *i.e.*, COS_t, is

$$COS_t = \sum_{t' \leq t} \sum_{r \in \mathcal{V}(G_A)} pos_{t'}(r). \tag{8}$$

For all search plans P, COS_T is the objective value, *i.e.*, $COS_T(P) = COS(P)$.

Table 1: The probability of containment for each vertex at each time step and the cumulative overall probability of success for each time step for the search plan P^* of the example of Figure 1. The probabilities are rounded to the third decimal.

	Probability of containment in vertex r at time t ($poc_t(r)$)													
t \ r	0	1	2	3	4	5	6	7	8	9	10	11	12	$COS_t(P^*)$
1	-	-	-	-	1	-	-	-	-	-	-	-	-	0
2	-	-	-	-	.500	-	-	.050	-	-	-	-	-	.450
3	-	-	-	-	.263	-	.012	.026	.012	-	-	-	-	.686
4	.001	.001	.001	.001	.138	.001	.008	.015	.013	.001	.001	.001	.001	.817
5	.001	.001	.001	.001	.073	.001	.008	.008	.01	.002	.002	.002	.002	.889

3 A Constraint Programming Model for the OSP

We present in this section the CP model and the heuristic we developed to guide the resolution process. We define the following constants:

– $\mathcal{T}$, the set of all time steps;
– $G_A = (\mathcal{V}(G_A), \mathcal{E}(G_A))$, the accessibility graph representing the search environment;
– $y_0 \in \mathcal{V}(G_A)$, the initial searcher's position;
– $poc_1(r)$, the initial probability of containment in vertex r ($\forall r \in \mathcal{V}(G_A)$);
– $pod(r)$, the conditional probability of detecting the object when $y_t = r$ ($\forall t \in \mathcal{T}, \forall r \in \mathcal{V}(G_A)$);

– $\mathbf{M}(s, r)$, the probability of an object's move from vertex s to vertex r in one time step ($\forall s, r \in \mathcal{V}(G_A)$).

Furthermore, we define $poc_t^{\text{Markov}}(r)$, the updated probability of containment in vertex r at time t in the absence of searches as:

$$poc_t^{\text{Markov}}(r) \stackrel{\text{def}}{=} \begin{cases} poc_1(r), & \text{if } t = 1; \\ \sum_{s \in \mathcal{V}(G_A)} \mathbf{M}(s, r) poc_{t-1}^{\text{Markov}}(s), & \text{otherwise.} \end{cases} \quad (9)$$

The Markovian probability of containment poc^{Markov} is an upper bound on the probability of containment in vertex r at time t, *i.e.*, $poc_t(r) \leq poc_t^{\text{Markov}}(r)$. This is due to the fact that an unsuccessful search in vertex r at time t decreases the probability of the object being there at time t (from Equation (5)). Moreover, we observe that the probability of success $pos_t(r)$ in vertex r at time t is bounded by the probability of detection in r ($pod(r)$), *i.e.*, $pos_t(r) \leq pod(r)$. Both of these observations will be used to bound the domains of the probability variables in the CP model.

3.1 The variables

The model's decision variables used to define the search plan P are:

– $Y_0 = y_0$, the initial searcher's position;
– $Y_t \in \mathcal{V}(G_A)$, the searcher's position at time t ($\forall t \in \mathcal{T}$).

The non-decision variables used to compute the COS criterion value are:

– $POC_1(r) = poc_1(r)$, the probability of containment in vertex r at time 1 ($\forall r \in \mathcal{V}(G_A)$);
– $POC_t(r) \in [0, poc_t^{\text{Markov}}(r)]$, the probability of containment in vertex r at time t ($\forall t \in \mathcal{T}, r \in \mathcal{V}(G_A)$) where $poc_t^{\text{Markov}}(r)$ is defined by Equation (9);
– $POS_t(r) \in [0, pod(r)]$, the probability of success in vertex r at time t ($\forall t \in \mathcal{T}, r \in \mathcal{V}(G_A)$);
– $COS \in [0, 1]$, the COS criterion value, *i.e.*, the sum of all local probabilities of success up to time T.

The domain of Y_t is finite ($\forall t \in \mathcal{T}$). The domains of the probability variables are infinite since these variables are real. Interval-valued domains are used to define these domains, *i.e.*, non-enumerated domains whose values are implicitly given by a lower bound and an upper bound.

3.2 The constraints

Constraint (10) defines the searcher's path, *i.e.*, the search plan P. It constrains the searcher to move from one vertex to another according to the accessibility graph edges $\mathcal{E}(G_A)$.

$$(Y_{t-1}, Y_t) \in \mathcal{E}(G_A), \qquad \forall t \in \mathcal{T}. \quad (10)$$

The constraints (11) to (13) compute the probabilities required to evaluate the COS criterion. The first two constraints, (11) and (12) compute the probability of success. Constraint (13) is the probability of containment update equation.

$$Y_t = r \implies POS_t(r) = POC_t(r)pod(r), \qquad \forall t \in \mathcal{T}, \forall r \in \mathcal{V}(G_A). \quad (11)$$

$$Y_t \neq r \implies POS_t(r) = 0.0, \qquad \forall t \in \mathcal{T}, \forall r \in \mathcal{V}(G_A). \quad (12)$$

$$POC_t(r) = \sum_{s \in \mathcal{V}(G_A)} \mathbf{M}(s,r) \left[POC_{t-1}(s) - POS_{t-1}(s) \right], \quad \forall t \in \{2, \ldots, T\}, \quad \forall r \in \mathcal{V}(G_A). \quad (13)$$

3.3 The objective function

We have experimented with two different encodings of the objective function. The first one encodes the objective function as a *sum*, the second one encodes it as a *max*. Both encodings are equivalent and lead to the same objective value.

The sum *objective function.* The sum encoding of Equation (14) consists of encoding the objective function as it appears in (6). It is the natural way to represent this function.

$$\max COS, \quad (14)$$

$$COS = \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{V}(G_A)} POS_t(r). \quad (15)$$

The max *objective function.* The sum constraint does a very poor job of filtering the variables: the upper bound on a sum of variables is given by the sum of the upper bounds of the variable domains. However, since we know that in the summation $\sum_r POS_t(r)$ only one variable is non-null, a tighter upper bound on this sum is given by the maximum domain upper bound. A tighter upper bound on the objective variable generally leads to a faster branch and bound. We therefore have implemented the objective function defined by Equation (6) using the following constraints:

$$\max COS, \quad (16)$$

$$COS = \sum_{t \in \mathcal{T}} \max_{r \in \mathcal{V}(G_A)} POS_t(r). \quad (17)$$

3.4 The proposed value selection heuristic

In this section we describe the value selection heuristic we developed. The idea of our heuristic is based on a stochastic generalization of a graph based pursuit evasion problem called the *cop and robber* game [17]; the description of its theoretical bases is beyond the scope of this paper. We were also inspired by

a domain ordering idea that was successfully used for the *multiple rectangular search areas* problem [12].

Our novel heuristic simplifies the probability system in the OSP problem by ignoring the negative information received by the searcher when s/he fails to detect the object. That is, at each time step $t \in \mathcal{T}$, the heuristic chooses the most promising accessible vertex based on the total probability of detecting the object in the remaining time. Therefore, we call our heuristic the *total detection* (TD) heuristic.

Let $G_A = \langle \mathcal{V}(G_A), \mathcal{E}(G_A) \rangle$ be the accessibility graph where the searcher and the object evolve. Let $t \in \mathcal{T}$ be a time step, and $y, o \in \mathcal{V}(G_A)$ the positions of the searcher and the object. Let $w_t(y, o)$ be the conditional probability that the searcher detects the object in the time period $[t, t+1, \ldots, T]$ given that, at time t, the searcher is in y and the object in o. The function $w_t(y, o)$ is recursively defined as follows:

$$w_t(y,o) \stackrel{\text{def}}{=} \begin{cases} pod(o), & \text{if } o = y \text{ and } t = T, \\ 0, & \text{if } o \neq y \text{ and } t = T, \\ pod(o) + (1 - pod(o))p_t(y,o), & \text{if } o = y \text{ and } t < T, \\ p_t(y,o), & \text{if } o \neq y \text{ and } t < T, \end{cases} \tag{18}$$

where

$$p_t(y,o) = \sum_{o' \in \mathcal{N}(o)} \mathbf{M}(o,o') \max_{y' \in \mathcal{N}(y)} w_{t+1}(y', o'), \tag{19}$$

is the probability of detecting the object in the period $[t+1, \ldots, T]$. Equations (18) and (19) have the following interpretation:

- If $t = T$, the searcher has a probability $pod(o)$ of detecting the object when the searcher and the object are co-located, *i.e.*, $o = y$; otherwise, the searcher and the object are not co-located and the probability is null.
- If $t < T$ and $o = y$, then the searcher can detect the object at time t with probability $pod(o)$ or fail to detect it at time t with probability $1 - pod(o)$. If the searcher fails to detect the object at time t, s/he may detect it during the period $[t+1, \ldots, T]$. The probability of detecting the object in the period $[t+1, \ldots, T]$ is given by $p_t(y, o)$ (Equation (19)) and may be interpreted as follows:
  - in the case where there is only one edge leaving vertex o to vertex o', the searcher chooses the accessible vertex y' that maximizes the conditional probability of detecting the object in the time period $[t+1, \ldots, T]$, given his/her new position y' and the new object's position o', *i.e.*, $\max_{y' \in \mathcal{N}(y)} w_{t+1}(y', o')$;
  - In the general case where vertex o has many neighbors, $p_t(y, o)$ is the average of all the maximal $w_{t+1}(y', o')$ weigthed by the probability $\mathbf{M}(o, o')$ of moving from o to o'.

  This is reasonable since we do not control the object's movements but we can move the searcher to the vertex that has the highest probability of success.

– Finally, if the search time is not over (*i.e.*, $t < T$) and the object and the searcher are not co-located (*i.e.*, $o \neq y$), the probability of detecting the object at time t is null and the probability of success depends entirely on the probability $p_t(y, o)$ of detecting the object within the period $[t+1, \ldots, T]$.

A searching strategy $S : \mathcal{T} \times \mathcal{V}(G_A)$ assigns to each time step and plausible searcher's position a set of vertices that are considered to be optimal according to some heuristic. In the TD heuristic case, the strategy sets the new searcher's position to be the accessible vertex that maximizes the probability of detecting the object in the remaining time:

$$S_t(Y_t) \stackrel{\text{def}}{=} \underset{y' \in \text{dom}(Y_t)}{\text{argmax}} \sum_{o \in \mathcal{V}(G_A)} w_t(y', o) poc_t(o), \qquad \forall t \in \mathcal{T}. \tag{20}$$

In order to apply this value selection heuristic, the following *static ordering* of the decision variables is used: $Y_0, \ldots, Y_T$. That is, the solver branches first on Y_0, then on Y_1 and so on. Each time the solver branches on a new path variable Y_t, the strategy $S_t(Y_t)$ is computed in polynomial time.

4 Experimentation

Our experiments were conducted in two phases. In Phase 1, we compared the two versions of the CP models presented in Section 3 (*i.e.*, CpMax and CpSum). In Phase 2, we examined the performance of the TD heuristic presented in Section 3.4 when used as a value selector along with the best CP model retained from Phase 1. The TD heuristic is compared with the CpMax model using an *increasing domain*[5] value selection heuristic. For all experiments, the following static ordering is used for branching: $Y_0, \ldots, Y_T$.

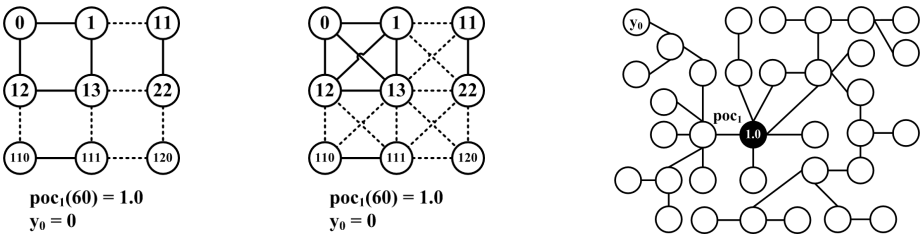

Fig. 2: The 11×11 grid G^+ (left), the 11×11 grid G^* (center), the graph G^L (right)

The graphs used in our benchmark along with the searcher's initial position y_0 and the initial probability of containment distribution poc_1 are shown on

[5] When branching on Y_t, the solver selects the integer with the smallest value.

Figure 2. G^+ is a reflexive 11×11 grid where all adjacent vertices except diagonals are linked by an edge. G^* is a reflexive 11×11 grid where all adjacent vertices (diagonals included) are linked by an edge. G^L is a reflexive graph generated using the Université Laval tunnels map. It is almost a tree. We tried these graphs with three different probabilities of detection: $pod(r) \in \{0.3, 0.6, 0.9\}$ $(\forall r \in \mathcal{V}(G_A))$. The assumed Markovian object's motion model is

$$\mathbf{M}(s,r) = \begin{cases} \frac{1-\rho}{\deg(s)-1}, & \text{if } (s,r) \in \mathcal{E}(G_A), \\ \rho, & \text{if } s = r, \end{cases} \tag{21}$$

where $\deg(s)$ is the degree of s and $\rho \in \{0.3, 0.6, 0.9\}$ is the probability that the object stays in its current location. The total times allowed for the searches are $T \in \{9, 11, 13, 15, 17, 19\}$. Usual OSP problem experiments use grids similar to G^+ (*e.g.*, [8], [10]). Therefore, our G^+ problem instances are comparable with those used in the literature.

All tests consisted of a single run on an instance $\left(G_A, T, pod(r)_{r\in\mathcal{V}(G_A)}, \rho\right)$, as described above. We allowed a total number of 5,000,000 backtracks and a time limit of 20 minutes. All implementations are done using Choco Solver 2.1.3 [18], a solver developed in the Java programming language, and the Java Universal Network/Graph (JUNG) 2.0.1 framework [19].[6] The probabilities of the OSP CP model were multiplied by an integer for implementation purposes. Because of numerical errors, our results are accurate to the fourth decimal. All tests were run on an Intel(R) Core(TM) i7-2600 CPU with 4 GB of RAM.

5 Results and discussion

In this section, we compare the time required to obtain various incumbent solutions (*i.e.*, the best feasible solutions found so far). The time to the last incumbent is the CPU time spent by the solver to obtain the incumbent with the best objective value within a 20 minutes or 5,000,000 backtracks limit.

5.1 Phase 1: Comparing the CP models.

Table 2 compares the results obtained with the CpMax model with the ones obtained with the CpSum model on a 11×11 G^+ grid with $T = 17$, various probability of detection values (pod) and various motion models (ρ). In all cases, the COS value of the last incumbent solution obtained with the CpMax model is higher or equal to the one obtained with the CpSum model. Furthermore, when there is a tie on the COS value, the time required with the CpMax model is lower than the one required with the CpSum model. The tendency of the CpMax model to outperform the CpSum model is present on G^* and on G^L instances with $T = 17$ (not shown). On most instances, the CpMax model requires fewer backtracks than the CpSum model to achieve a higher quality last incumbent solution. We

[6] The source code of our experiments is available upon request.

conclude that the use of the constraint "max" leads to a stronger filtering on the variable COS, and thus, computes a tighter bound on the objective function. For this reason, further comparisons involve only the CpMax model.

Table 2: The COS value of the last incumbent solution on a 11×11 G^+ grid with $T = 17$. Bold font is used to highlight the best objective value (higher is better). Ties are broken using the time to last incumbent value.

		CpMax		CpSum	
$pod(r)$	ρ	Time to last incumbent (s)	COS value of the last incumbent	Time to last incumbent (s)	COS value of the last incumbent
0.3	0.3	**1197.15**	**0.0837**	1045.66	0.0831
	0.6	**1198.56**	**0.1276**	990.61	0.1267
	0.9	**1026.02**	**0.3379**	1165.88	0.3379
0.6	0.3	**959.18**	**0.1532**	999.45	0.1532
	0.6	**1168.98**	**0.2202**	1015.64	0.2172
	0.9	**1166.29**	**0.5122**	942.36	0.5014
0.9	0.3	**1161.59**	**0.2162**	1184.86	0.2162
	0.6	**692.16**	**0.3151**	727.57	0.3151
	0.9	**1169.91**	**0.6283**	879.59	0.6252

5.2 Phase 2: Evaluating the TD value selection heuristic.

Figures 3 shows the COS value as a function of time (ms) obtained on the G^+, G^* and G^L environments with $pod(y_t) = 0.6$ ($\forall t \in \mathcal{T}$) and a motion model such that the probability ρ that the object stays in its current location equals 0.6. On all instances shown, the benefits of using the TD heuristic as a value selection heuristic are clear as the solver finds incumbent solutions of higher quality in less time when compared to the CpMax model using an increasing domain value selection heuristic. In all cases shown, the COS value of the first incumbent solution found with the TDValSel+CpMax configuration, a solution encountered after less than 1 second of solving time, is within 5% of the COS value of the last incumbent solution. On the G^+ instance with $T = 17$, TDValSel+CpMax encountered 21 solutions before settling to an incumbent with $COS = 0.2978$ while the CpMax configuration encountered 98 solutions before settling to an incumbent with $COS = 0.2202$. On the G^* instance with $T = 17$, TDValSel+CpMax encountered 46 solutions before settling to an incumbent with $COS = 0.3478$ while the CPMax configuration encountered 70 solutions before settling to an incumbent with $COS = 0.2959$. Finally, on the G^L instance with $T = 17$, TDValSel+CpMax encountered 29 solutions before settling to an incumbent with $COS = 0.7930$ while the CPMax configuration encountered 23 solutions before settling to an incumbent with $COS = 0.6676$. By looking at the total number of

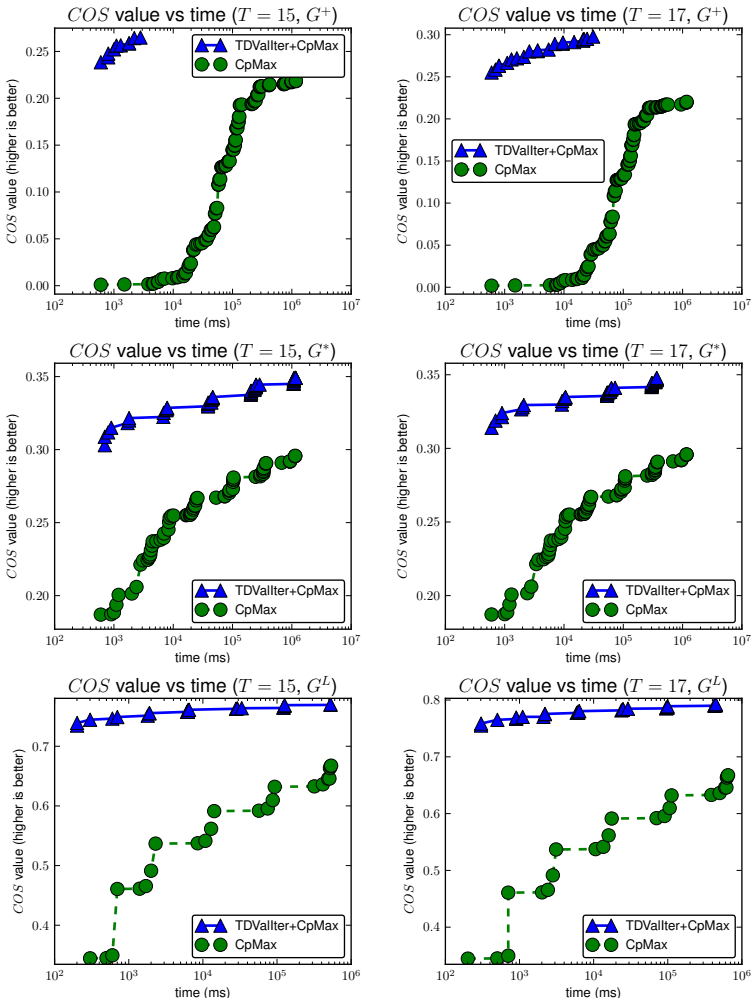

Fig. 3: The COS value as a function of time (ms) (log scale) obtained with the TDValSel+CpMax and the CpMax configurations on a 11×11 G^+, on a 11×11 G^* instance and on a G^L instance with $T = 15$ (left column) and $T = 17$ (right column). The $pod(y_t) = 0.6$ ($\forall t \in \mathcal{T}$), and the $\rho = 0.6$.

solutions encountered by the two configurations on the three instances, it seems that varying the graph structure leads to very different problem instances. This is partly due to the motion model of the object and to the probability of staying in place ρ: Given an object's position r, the remaining probability mass $1 - \rho$ is distributed among the neighbors of r leading to smaller probability of containment poc values in the neighborhood of r on G^* and G^+ than on most vertices

Table 3: The COS value of the last incumbent solution on a 11×11 G^+ grid with $T = 17$. Bold font is used to highlight the best objective value (higher is better). Ties are broken using the time to last incumbent value.

		TDValSel+CpMax		CpMax	
$pod(r)$	ρ	Time to last incumbent (s)	COS value of the last incumbent	Time to last incumbent (s)	COS value of the last incumbent
0.3	0.3	**5.31**	**0.1055**	1197.15	0.0837
	0.6	**19.42**	**0.1645**	1198.56	0.1276
	0.9	**3.70**	**0.4418**	1026.02	0.3379
0.6	0.3	**159.42**	**0.1893**	959.18	0.1532
	0.6	**31.02**	**0.2978**	1168.98	0.2202
	0.9	**225.66**	**0.6559**	1166.29	0.5122
0.9	0.3	**54.94**	**0.2595**	1161.59	0.2162
	0.6	**37.73**	**0.4119**	692.16	0.3151
	0.9	**467.34**	**0.8194**	1169.91	0.6283

of G^L. For this reason, G^* and G^+ are significantly harder instances to solve than G^L. Furthermore, the G^+ and the G^* accessibility graphs involve more symmetric instances than G^L.

Table 3 compares the results obtained with the TDValSel+CpMax configuration to the ones obtained with the CpMax model using an increasing domain value selection heuristic on a 11×11 G^+ grid with $T = 17$, various probability of detection values (pod), and various motion models (ρ). Again, the TDValSel heuristic is dominant with a time to last incumbent up to 300 times faster for a higher quality solution in terms of COS value. The tendency of the TDValSel+CpMax configuration to outperform the CpMax model using an increasing value selection heuristic is present on G^* and on G^L instances with $T = 17$ as well. For this reason, the tables for the G^* and the G^L instances are omitted.

Figure 4 compares the COS values obtained on several G^+ 11×11 grid with $pod(y_t) = 0.6$ ($\forall t \in \mathcal{T}$), and $\rho = 0.6$ instances of increasing complexity in T. For all values of T and instance types, the COS value of the last incumbent found with the TDValSel+CpMax configuration is higher or equal to the one found with the CpMax model alone. On the G^+ instance, we notice that the solution is found in less than 5 seconds up to $T = 17$. By comparing the first row of figure 4 showing the G^+ instance to the other rows, we notice that time to last incumbent curve of the TDValSel+CpMax is more erratic on the G^* and the G^L instances. We believe that this may be due to the precision we used to compute the probability variables (a limitation of our solver).

In order to get an idea of the relative performance of our model and value selection heuristic, we compared our results with results published in the literature using a BB algorithm [10]. After communications with the authors of [10] we were able to validate that our solutions for the G^+ instances are optimal up

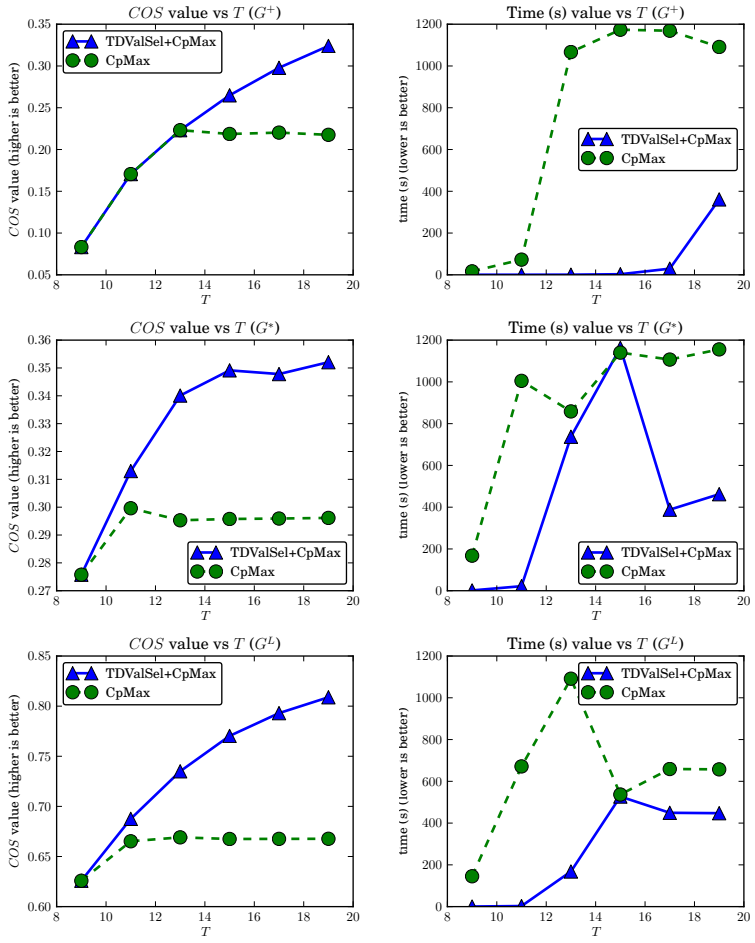

Fig. 4: The COS value (left) and the time to last incumbent (ms) (right) as a function of the total number of time steps (T) with the TDValSel+CpMax and the CpMax configurations on a G^+, a G^* and a G^L instance where $pod(y_t) = 0.6$ ($\forall t \in \mathcal{T}$), and $\rho = 0.6$.

to the fourth decimal. However, the instances are too large for our solver to prove optimality in a reasonable time. Table 4 presents the time to last incumbent on a 11×11 G^+ grid with $pod(r) = 0.6$ and $\rho = 0.6$, and the time spent by a BB procedure to prove the optimality of its last incumbent solution when using various bounds from the literature. The hardware and software configurations used to produce these results differ from ours. Consequently, the goal of this comparison is simply to show the general tendency on the instances for which the optimal value is published rather than proving that our approach outper-

Table 4: The time to last incumbent on a 11×11 G^+ grid with $pod(r) = 0.6$ and $\rho = 0.6$ compared to the time spent by a BB procedure to prove optimality when using various bounds [10].

	Time to incumbent (s)	Time to optimality (s)*			
T	TDValSel+CpMax	DMEAN	MEAN	PROP	FABC
15	2.80	3.14	12.27	8.16	62.64
17	31.02	23.76	71.57	37.20	352.96
**The time values are taken from [10].* *They are used to give a general idea of how our results behave.*					

forms the BB procedure. Recalling that we are not using any problem specific bound on the objective function (except a simple objective function simplification carried out in the CpMax model), our results, comparable to the ones in the literature, highlight the performance of the TD value selection heuristic.

One of the main advantage of using constraint programming is expressivity. The constraint programming model stays close to the natural problem formulation while enabling strong filtering (*e.g.*, the CpMax model) and heuristics (*e.g.*, the TD value selection heuristic). We believe that using a CP model is closer to the natural problem formulation than an IP model for example. In addition, the model can be easily adapted and extended. For instance, the model and the heuristic can be generalized to allow searches from a distance, *i.e.*, the searcher sees a subset of visible vertices including his position.

6 Conclusion

We have presented a CP model to solve the OSP problem. This model includes a very efficient value selection heuristic that branches on vertices leading to a high objective value (*i.e.*, probability of success). We refined the objective function to obtain a tighter bound on the objective value without discarding solutions. Experiments show that our model is competitive with the state-of-the-art in search theory and that constraint programming is a good technique to solve the OSP. Future work includes the development of tight bounds in order to allow the solver to prove the optimality of its incumbent solution. We believe that such a bound could be based on the information already computed for the value selection heuristic presented in this paper.

Acknowledgments

We would like to thank Haye Lau for his help in validating the results of Section 5.2, and the anonymous reviewers for their helpful comments and suggestions.

References

1. K. Trummel and J. Weisinger, "The complexity of the optimal searcher path problem," *Operations Research*, vol. 34, no. 2, pp. 324–327, 1986.
2. L. Stone, *Theory of Optimal Search*. New York: Academic Press, 2004.
3. E. Kranakis, D. Krizanc, and S. Rajsbaum, "Computing with mobile agents in distributed networks," in *Handbook of Parallel Computing: Models, Algorithms, and Applications* (S. Rajasedaran and J. Reif, eds.), pp. 1–26, CRC Press, 2007.
4. R. Chandramouli, "Web search steganalysis: Some challenges and approaches," in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 576–579, 2004.
5. M. Verkama, "Optimal paging - a search theory approach," in *Proceedings of the international conference on universal personal communications*, pp. 956–960, 1996.
6. T. Stewart, "Search for a moving target when the searcher motion is restricted," *Computers and Operations Research*, vol. 6, pp. 129–140, 1979.
7. J. Eagle, "The optimal search for a moving target when the search path is constrained," *Operations Research*, vol. 32, no. 5, pp. 1107–1115, 1984.
8. J. Eagle and J. Yee, "An optimal branch-and-bound procedure for the constrained path, moving target search problem," *Naval Research Logistics*, vol. 38, no. 1, pp. 110–114, 1990.
9. A. Washburn, "Branch and bound methods for a search problem," *Naval Research Logistics*, vol. 45, pp. 243–257, 1998.
10. H. Lau, S. Huang, and G. Dissanayake, "Discounted mean bound for the optimal searcher path problem with non-uniform travel times," *European journal of operational research*, vol. 190, no. 2, pp. 383–397, 2008.
11. G. Martins, "A new branch-and-bound procedure for computing optimal search paths.," tech. rep., Naval Postgraduate School, 1993.
12. I. Abi-Zeid, O. Nilo, and L. Lamontagne, "Resource allocation algorithms for planning search and rescue operations," *INFOR*, vol. 49, no. 1, pp. 15–30, 2011.
13. K. M. Brown and I. Miguel, "Uncertainty and change," in *Handbook of Constraint Programming* (F. Rossi, P. V. Beek, and T. Walsh, eds.), pp. 44–58, Springer, 2006.
14. S. Tarim, S. Manandhar, and T. Walsh, "Stochastic constraint programming: A scenario-based approach," *Constraints*, vol. 11, no. 1, pp. 53–80, 2006.
15. S. Brown, "Optimal search for a moving target in discrete time and space," *Operations Research*, vol. 28, no. 6, pp. 1275–1289, 1980.
16. G. Verfaillie and N. Jussien, "Constraint solving in uncertain and dynamic environments: A survey," *Constraints*, no. 10, pp. 253–281, 2005.
17. R. Nowakowski and P. Winkler, "Vertex-to-vertex pursuit in a graph," *Discrete Mathematics*, vol. 2-3, no. 43, pp. 235–239, 1983.
18. F. Laburthe and N. Jussien, *Choco Solver Documentation*, 2012. `http://www.emn.fr/z-info/choco-solver/`.
19. J. O'Madadhain, D. Fisher, T. Nelson, S. White, and Y. Boey, "Jung: Java universal network/graph framework." `http://jung.sourceforge.net`, 2010.