

# CORALS: A Real-Time Planner for Anti-Air Defence Operations<sup>1</sup>

ABDER REZAK BENASKEUR

Defence R&D Canada

FRODUALD KABANZA

Menya Solutions

and

ERIC BEAUDRY

Menya Solutions

---

Forces involved in modern conflicts may be exposed to a variety of threats, including coordinated raids of advanced ballistic and cruise missiles. To respond to these, a defending force will rely on a set of combat resources. Determining an efficient allocation and coordinated use of these resources, particularly in the case of multiple simultaneous attacks, is a very complex decision-making process, in which a huge amount of data must be dealt with, under uncertainty and time pressure. This paper presents **CORALS** (**C**ombat **R**esource **A**llocation **S**upport), a real-time planner developed to support the command team of a naval force defending against multiple simultaneous threats. In response to such multiple threats, CORALS uses a local planner to generate a set of local plans, one for each threat considered apart, and then combines and coordinates them into a single optimized, conflict-free global plan. The coordination is performed through an iterative process of plan merging and conflict detection and resolution, which acts as a plan repair mechanism. Such an incremental plan repair approach also allows adapting previously generated plans to account for dynamic changes in the tactical situation.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation; Heuristic methods; Scheduling*; J.7 [**Computer Applications**]: Computers in Other Systems—*Command and control; Military; Real Time; Consumer products*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Planning, Decision Support, Anti-Air Defense Operations.

---

---

<sup>1</sup>This paper is an extended and enhanced version of [Benaskeur et al. 2008].

---

Author's address: A. Benaskeur, Defence R&D Canada, Québec (QC) G3J 1X5, Canada

abderrezak.benaskeur@drdc-rddc.gc.ca

F. Kabanza and E. Beaudry, Menya Solutions, Sherbrooke (QC) J1H 47B, Canada,

{kabanza,eric.beaudry}@menyasolutions.com

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2010 ACM 1529-3785/2010/0700-0001 \$5.00

## 1. INTRODUCTION

Modern military forces operate in a large variety of situations with constantly increasing complexity. To cope with diverse threats, referred to as the attackers in the sequel, a defending force, either operating alone or in consort with partners, will need to efficiently plan and coordinate the use of its defensive combat resources, under very short reaction time constraints, on the basis of imperfect information. Determining an efficient allocation and coordination of the combat resources is a very complex decision-making problem, referred to, herein, as Combat Power Management (CPM). Different studies have shown that, faced with the CPM problem, experienced military commanders are competent in responding efficiently to multiple sequential, and sufficiently spaced, threats. However, in situations involving massive raids, with coordinated threats arriving in quick successions, and from different directions, the quality of human decisions tends to deteriorate with an increasing number of threats [Ousborne 1993; Bos et al. 2005].

In its general form, the CPM problem consists in: (i) generating a plan to defeat threats; (ii) executing the plan by engaging the threats, (iii) monitoring the execution of the plan to detect, assess and handle contingencies; and (iv) assessing the outcome of executed actions (*i.e.*, assessing the damage inflicted to engaged threats to determine to what extent the engagement objectives were attained). In this context, an action is defined as the application of some combat resource in a single step (*e.g.*, firing a missile against a threat; or launching a decoy).

The CPM problem is mainly characterized by durative concurrent actions having time-dependent probabilistic effects. These actions compete for limited combat resources, which are of different types and may have different effects and effectiveness. In general, the combat resources include hard-kill (or lethal) weapons (*e.g.*, missiles and guns), soft-kill (or non-lethal) weapons (*e.g.*, electronic counter measures), deterrence measures, and the maneuvering of the defending force.

The planning of actions in the context of CPM is a complex problem for force commanders, particularly in situations involving multiple simultaneous threats, arriving from different directions. The complexity arises from a combination of factors, including the following [Bos et al. 2005; Ousborne 1993]:

- Some of the key parameters, such as target identity and localization, required for determining the use of combat resources are subject to uncertainty.
- Combat resources have different probabilities of successfully defeating threats depending on a variety of conditions. Moreover, combinations of combat resources can have different probabilities of effectiveness against targets. Even a single target may have to be engaged by coordinated combat resources to maximize the probability of defeating it.
- There are often different kinds of constraints on the use of combat resources. In particular, weapon types are effective within their specific ranges and soft-kill decoys remain in the air for specific time windows.
- Other factors, such as deadlines for engaging targets (targets do not wait politely for decisions to be made), may contribute to the complexity of the problem.

This paper presents CORALS (COmbat Resource ALlocation Support), a real-time planner developed to support a defending force command team during the

conduct of CPM-related activities, by automatically planning actions against multiple threats, arriving from different directions. While the basics of CORALS are outlined in [Benaskeur et al. 2008], this paper provides more details about the algorithms and their performance.

Although the motivation of the development of CORALS has been driven by a requirement to develop an Anti-Ship Missile Defence capability prototype for naval warfare operations, the discussion here will be kept as independent as possible from the specificities of the application domain, and more importantly will be kept unclassified. Moreover, the generic implementation of CORALS makes it suitable for any domain that deals with planning with resources under constraints. CORALS is an activity-oriented planner, which solves planning problems that are defined by formulating related activities and an objective function conveying the goal to be achieved by the activities. When applied to the specific CPM domain, activities correspond to targets and the goal to maximization of the probability of raid annihilation, which is built up the probabilities of the successful engagement of the different targets.

Section 2 discusses work related to planning and the CPM domain. The latter is introduced in Section 3. Section 4 presents the high-level architecture of CORALS. The details of the different CORALS algorithms are discussed in Section 5. Then the evaluation results of CORALS are presented and discussed in Section 6, and concluding remarks with some perspectives on future works are given in Section 7.

## 2. RELATED WORK

The CPM problem is often referred to as the Weapons Assignment problem. The survey of the literature showed that none of the existing approaches comprehensively addresses all aspects of the CPM, since they all make different simplifying assumptions. Different approaches have different virtues and limitations, depending on the assumptions they make to simplify the problem and the algorithms they use to solve it. The vast majority of approaches considers formulations of the Weapons Assignment problem that consists only in pairing weapons with targets, abstracting away other aspects of the problem [Benaskeur et al. 2008]. In particular, many Weapons Assignment formulations do not consider scheduling aspects (*i.e.*, the Weapons Assignment does not involve assigning actions over time periods); others do not explicitly handle interferences among actions. Furthermore, most approaches do not consider time constraints or actions undertaken against targets in a sequence over a given time horizon. The CPM problem is not just about deciding which combat resources should engage which target, but also concerns the sequential decision-making about when actions against the targets should best be executed and how they should be coordinated, with past, current and future actions, in order to minimize negative interferences and maximize positive interferences.

There are generally two main formulations of the CPM problem in the literature: episodic (also called static in some reference) and sequential (also referred to as dynamic by some authors). The episodic formulation consists in finding, in single step, pairings weapons/targets that maximize the expected survival value of the defending force, while the sequential formulation considers the dynamic and time-dependence nature of the decision-making problem. Even under the ‘simpli-

fighting' episodic assumption, the problem is known to be NP-complete [Lloyd and Witsenhausen 1986].

A survey of early Operational Research approaches, back in the 1970s, is given in [Matlin 1970]. Most of these approaches were based on linear integer programming, with few considering heuristic search. More recent approaches still use the same basic techniques, but rely on enhanced implementations [Ahuja et al. 2003; Bohachevsky and Johnson 1993; Hadj-Alouane et al. 1999]. The fact that the Weapons Assignment problem has been, to date, mostly studied in the Operational Research literature is not surprising, since dealing with resources and durative actions is very common in the Operational Research literature. There exist efficient scheduling algorithms that use heuristics to maximize various types of objective functions other than makespan [Brucker and Knust 2006]. However, Operational Research approaches have been essentially limited to the episodic formulation of the (Weapons) Assignment problem.

The very few exceptions that considered formally the sequential aspect of the CPM problems include [Hosein and Athans 1990]. However, the authors did not address durative actions. Other approaches have been proposed to solve the problem as a Markov Decision Process [Bertsekas et al. 1999; Meuleau et al. 1998], but do not handle durative actions either.

An in-depth analysis of the CPM problem suggested that a key element to solving it efficiently would be to reason explicitly about actions –that is, about the pre-conditions and effects of actions and about how actions interfere with each other. Reasoning about actions is more a feature of Artificial Intelligence planning approaches than Operational Research approaches. There exist few planners in the Artificial Intelligence planning literature that can handle concurrent durative actions with probabilistic effects, such those characterizing the CPM domain. Three of the most renowned are Tempastic [Younes and Simmons 2004], CPTP [Mausam and Weld 2006], and FPG [Buffet and Aberdeen 2006]. Tempastic follows an event-based modeling that does not naturally fit the CPM domain. CPTP is limited to a small number of actions. A requirement, imposed by the naval operational community, that CORALS has to satisfy, and that none of these planner does, is the ability to merge plans generated by an independent planner.

### 3. THE COMBAT POWER MANAGEMENT DOMAIN

#### 3.1 State Variables

A **state** in the CPM domain is given by the composition and position of the defending force, with the availability and the status of its defensive combat resources, the list of targets including their characteristics (range, bearing, speed, and identification), and the relevant parameters of the environment, such as humidity and wind direction and velocity. Figure 1 illustrates an example of an initial state where three targets (Target<sub>1</sub>, Target<sub>2</sub>, Target<sub>3</sub>) are attacking a defending force.

A cognitive work analysis of how military commanders plan against air targets showed that they generally abstract away pretty much the uncertainty characterizing the target kinematics [Bos et al. 2005]. They plan assuming accurate knowledge about the target and handle uncertainty during execution monitoring and re-planning. However, uncertainty related to action outcomes (discussed in

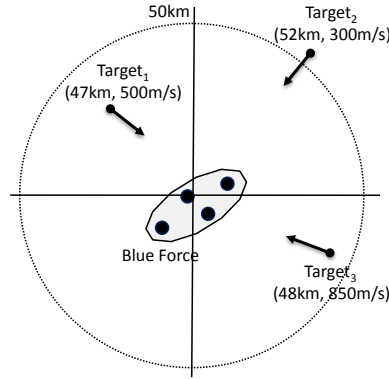


Fig. 1. A state in the CPM domain

Section 3.4) is directly accounted for during the planning phase. The CORALS planner follows a similar approach, by assuming a completely defined state, rather than a belief state which would explicitly account for the uncertainty in the state. Adopting a planning approach that matches the way military experts generate plans may not necessarily be the most efficient approach. However, this was a key argument in convincing military operational community of the relevance of automated planning-based decision support tools in a domain as critical as a naval warfare.

### 3.2 Resources

Combat resources, as any other type of resources, can be consumable, *i.e.*, each action consumes a fixed amount out of a limited quantity, or renewable, *i.e.*, they are temporarily unavailable during their usage, but become available once released. In military operations, defensive resources are often classified into two main categories: hard-kill (or lethal) resources and soft-kill (or non-lethal) resources. Hard-kill resources are used to intercept their targets and actively destroy them through direct impact or explosive detonation in the proximity. These, for instance, may include Surface-to-Air Missile (SAM) launchers, and/or various types of guns. Soft-kill resources use various techniques to deceive or disorient their targets to cause them to destroy themselves, or at least to lose their lock on the defending force. Example of soft-kill resources are given by decoys, which are used to seduce or distract targets, and electronic countermeasures, which are used to perturb the attackers use of the electromagnetic spectrum (*e.g.*, through the jamming of the radars).

The deployment of hard-kill and soft-kill, seen as primary combat resources, may require support from other resources considered as secondary. For instance, Fire Control Radars (FCRs) are required to track and illuminate the targets for the guidance of the missiles and the pointing of the guns. Another secondary combat resource is given by the movement and the positing of the defending force. These may be required for an effective deployment of the hard-kill and/or soft-kill primary combat resources. The defending force movement and positing can also be used as a primary combat resource, independently of hard-kill and soft-kill deployments, such as to minimize own detectability, or to reduce own vulnerability. The defending force can, in addition, use deterrence measures (*e.g.*, warning it by radio or illumi-

nating it with Fire Control Radars) as a combat resource to dissuade the attackers from taking action. While the launchers, the Fire Control Radars, deterrence, and the force movement and positing are considered as renewable resources, missiles, decoys and gun rounds are consumable resources, considered non-renewable for the duration of the operation.

A combat resource is specified by describing the types of targets it can be used against, its expected performance and the constraints on its operations, as discussed in the next subsections.

### 3.3 Constraints

There are different kinds of constraints on the deployment of combat resources. For instance, the number of launchers for missiles or decoys and the number of guns are generally very limited. Resources also have different ranges (*i.e.*, distances beyond which it is ineffective against specific targets), engagement geometry, with the possibility of blind zones (*i.e.*, angular sectors where a combat resource is not deployable), reload time (*e.g.*, the time required to reload a gun), and repositioning speed (*i.e.*, the speed at which a resource is repositioned against a new target coming from a different direction).

Consumable resources are also generally available in limited quantities, for the duration of the operations. Some resources must be used in combination with others (*e.g.*, when a Surface-to-Air Missile is launched, there must be a Fire Control Radar illuminating the target during the missile flight). Hence, the number of concurrent engagements by missiles is constrained both by the number of available missile launchers and the number of available supporting Fire Control Radars.

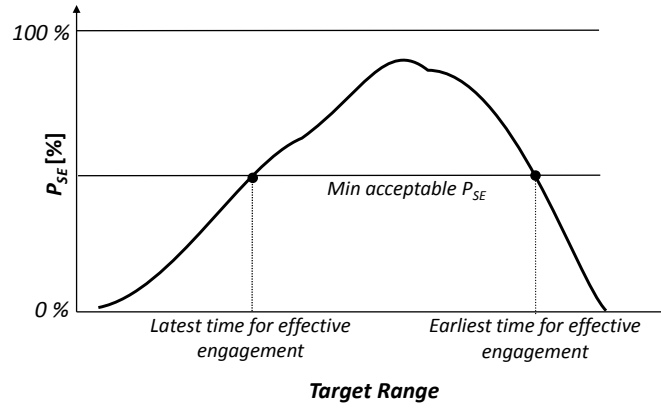
### 3.4 Actions

An **action** is defined here as the application of one or a combination of defensive combat resources against a single target. For instance, firing a missile against a target is considered as an action. Assigning a Fire Control Radar to track and illuminate a target is different action. Alternatively, one could model both actions as one of firing a missile, which requires both the missile launcher and the Fire Control Radar resources. For this particular case, conflicts are better handled by separating them because in general the missile and the supporting Fire Control Radar are not deployed simultaneously; their deployments rather overlap.

To each action is associated a **probability of successful completion ( $P_{SC}$ )**. Two different  $P_{SC}$  are used here: **probability of successful engagement ( $P_{SE}$ )** for individual combat resources and plans against individual targets, and **probability of raid annihilation ( $P_{RA}$ )** for the defending force survival. For a given combat resource, the  $P_{SE}$  may depend on several factors, including the range at which the interception takes place, the type and the characteristics of the engaged target, and the conditions of the surrounding environment. Figure 2 shows an example of an unclassified  $P_{SE}$ , as a function of the range at which the target is intercepted.

The small vertical dashed lines show the minimum and maximum range for an effective engagement. Other factors besides the range may affect the action outcome (and the  $P_{SE}$  curve). Those factors are not modeled explicitly and are treated as contingencies during plan execution and monitoring.

To illustrate, the action ‘Fire-SAM’ and the problem definition are specified in

Fig. 2. Example of combat resource  $P_{SE}$ 

the **Planning Domain Definition Language (PDDL)** [Fox and Long 2003] as follows.

```
(:domain
(:durative-action Fire-SAM
:parameter(?target - t)
:duration(=
  (/ (-(detectrange t) (*(starttime)(speed t)))
    (+ (speed t) 900)))
:condition(and
  (at start (not (killed t))))
:effect(and
  (at starttime (consume (AvailableSAM) 1))
  (during[starttime,endtime] (reserve (FCR) 1)))
:probabilistic-effect(
  (:probability (call SAM-Killprobability (t
    (-(detectrange t)*(endtime)(speed t))))))
  (:effect (killed t)))
...))
(:problem
(:objects T1 T2 T3 - target)
(:init
  (= (detectrange T1) 52000)
  (= (speed T1) 300)
  ...)
(:goal (and
  (killed T1) (killed T2) (killed T3))))
```

The domain expert specifies the preconditions, resources utilization and the expected probabilistic effects. For the Fire-SAM action, the duration is computed by estimating the interception time that depends on the target detection range, the action starting time, the target speed and the Surface-to-Air Missile speed (assumed 900 m/s). Since the  $P_{SE}$  depends on the interception range, a table that

summarizes the combat resource  $P_{SE}$  were provided by the domain experts.

### 3.5 Goals

The **goal** is defined by an objective function to optimize. For instance, in the CPM domain, a goal may be to maximize the probability of destroying all the targets. Alternatively, given the criticality for each target, the goal may be to minimize the overall expected damage to the defending force. Furthermore, by defining values for different areas of potential impact and vulnerability, the goal may be to maximize the expected survival value of the defending force. Additionally, one may want to optimize the use of available resources as a minimization of the monetary or casualty-related cost of the battle, or as resource conservation for future engagements in a sequential formulation of the problem. The goal can also be a combination of all the previous functions.

### 3.6 Activities and Plans

The concept of CORALS planner is centered on the notion of activity. An **activity** is defined as the execution of a plan. Different activities may interact with each others to achieve a common goal. In the CPM domain, each target has an activity associated with it that aims to neutralize it. Associated with each activity is a plan, which is composed of ordered (times) list of actions and uses a set of combat resources. Figure 3 illustrates a plan that maximizes the survival probability of defending naval warship. The plan corresponds to the tactical situation depicted in Figure 1.

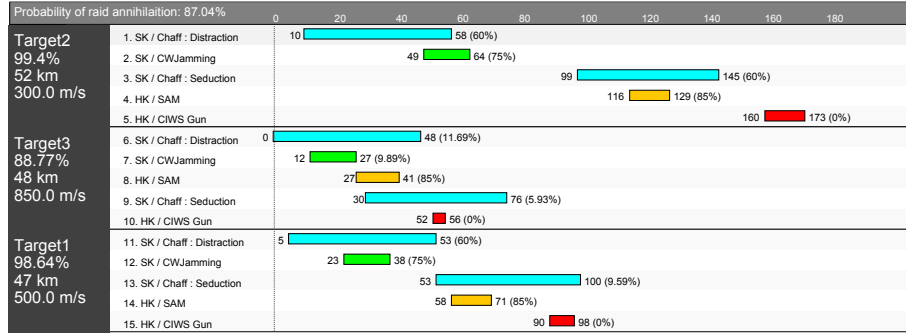


Fig. 3. Example of engagement plan

Actions are grouped by activities (targets) and represented by solid bars on the time line. A plan against a single target is also called a **local plan**. A **global plan** is defined as a combination of several local plans and concerns several simultaneous activities. Global plan allows countering more than one target simultaneously. The distinction between local plans and global plans may appear as superfluous, but it is of prime importance from the application domain perspective. In naval domain, for instance, a clear distinction is made between Anti-Ship Missile Defence problem, which consists in defending against a single target, and Distributed Anti-Ship Missile Defence, which consists in defending against multiple simultaneous



targets. Anti-Ship Missile Defence and Distributed Anti-Ship Missile Defence are considered as two different classes of problems by naval warfare experts, which justify the distinction made between the two in CORALS.

Figure 3 shows local plans, respectively for three Anti-Ship Missile targets: Target<sub>1</sub>, Target<sub>2</sub>, and Target<sub>3</sub>, considered of different types. The interpretation of the first plan is as follows: 10 seconds after the engagement begins, a decoy (Chaff) is launched to distract the target (avoid detection of the defending ship by the target) and will remain effective until time 58. At 116 seconds, a Surface-to-Air Missile is launched and is expected to intercept the Target<sub>1</sub> at time 129. The two other local plans can be interpreted in a similar way.

Note that contingencies are implicitly represented in the list of actions. When the current action in a local plan fails to neutralize the corresponding target, the next action is executed. Once the target is neutralized, subsequent actions are not executed, and the associated combat resources become available so the planner can consider using them against other targets during the re-planning phase. Each action increases the cumulative success probability of the plan. Actions in Figure 3 are displayed with the corresponding  $P_{SE}$ . The cumulative  $P_{SE}$  of the global plan, called the Probability of Raid Annihilation ( $P_{RA}$ ), is displayed on the first line. The  $P_{RA}$  represents the defending force's overall survival probability to all the targets. This probability is given under the assumption that a non-destroyed target is sufficient to neutralize the defending force (e.g., sink the ship). As mentioned before, the planner allows for the use alternative objective functions, including those taking into account partial damage to the defending force by optimizing its survival value.

This plan graphical representation, specific to the CPM domain, was dictated by requirements derived by Human-Computer Interaction experts. It was also validated with naval warfare experts. The Human-Computer Interaction experts discard the explicit representation of branching contingencies as they found them more difficult to digest in situations requiring short reaction times. For different application domains that may require a branching representation, CORALS internal structure allows for the addition of conditional branches, for instance, to handle the likelihood of new targets appearing in the theatre of operations. A further discussion of the graphical representation requirements is beyond the scope of this paper.

### 3.7 Conflicts

A **conflict** is a violation of one or more constraints by a subset of actions in the plan. An example is given by a plan that contains three overlapping missile launch actions, each against a different target. Since the model of the ship used in the simulation on only has two Fire Control Radar channels and a Fire Control Radar is required for the guidance of each missile, only a subset of the actions in the plan will be feasible. Such a plan is considered as having conflicts. The subset of actions causing a conflict is referred to as conflicting actions. We also call “conflict” the set of conflicting actions. A constraint violation caused by just one action (i.e., a subset of actions that is a singleton) is also a conflict. An example is an action using a combat resource in its blind zone. This is somewhat language abuse adopted for the sake of simplicity; otherwise an action cannot be in conflict with itself.

#### 4. CORALS ARCHITECTURE

Figure 4 shows the CORALS high-level architecture. CORALS takes as input a list of activities (or targets) with corresponding features, the current state of the defending force, and the environmental parameters. It returns an optimal global plan for all the activities and the associated estimated probability of successful completion ( $P_{SC}$ ). The domain knowledge consists of the description of resources and the specification of actions. CORALS proceeds in three steps, as described in the following sections.

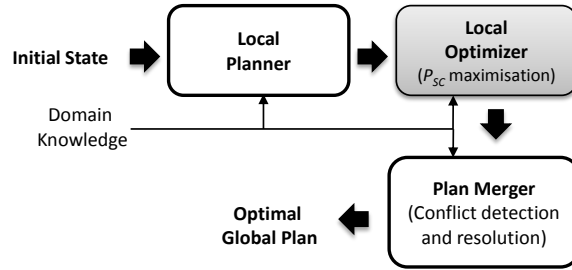


Fig. 4. CORALS architecture

##### 4.1 Local Planning

For each of the input activities, CORALS calls a local planner to obtain a local plan. This consists in generating plan against individual targets, taken separately. Current implementation of CORALS uses an internal local planner, but also exploits an external stand-alone provided military operational community. The latter involves a more accurate modeling for a single-target engagement, although, from the purely algorithmic perspective, the CORALS internal local planner is superior.

##### 4.2 Local Optimization

The generated local plans are optimized individually to maximize their  $P_{SC}$  against their respective target.

##### 4.3 Plan Merging

Plan merging consists, in response to multiple simultaneous or sequential threats, to merge the local plans generated and optimized separately into a single global plan, then to iteratively detect and resolve conflicts using a plan repair approach. CORALS plan repair performs search through a space of global plans. The use of a plan repair approach also allows revising previously generated plans to account for dynamic changes in the tactical situation.

The plan merging is the key component of the CORALS architecture, and is performed using a search through a space of global plans. The plan space exploration process can also be understood as a plan-repair approach given that it proceeds by repairing flaws (conflicts) in the partial plan by applying a sequence of plan transformations. Starting with a possibly conflicting global plan that is a union of

a set of local plans, the search process iteratively repairs the global plan by applying a sequence of transformations. A plan transformation consists in detecting and resolving conflicts in the current plan. As there are a huge number of possible plan transformation sequences, the key challenge in implementing such approach resides in the design of efficient data structures and algorithms for representing and detecting conflicts. Another important challenge consists in defining good heuristics that determine the order in which different transformations should be examined.

The CORALS underlying plan repair approach is reminiscent of the approach used by Automated Scheduling and Planning ENvironment (ASPEN) [Estlin et al. 1999]. Nonetheless, there are few key differences between the two planners. ASPEN uses Temporal Constraint Networks to represent temporal relationship between actions (modeled as activities) and to detect time conflicts, which consists in this case, in checking whether temporal constraints currently imposed among activities are consistent. Non-time-related dependencies, defined in CORALS as resource-related constraints, are represented using a Parameter Dependency Network, which records dependencies among activity parameters. Instead of Temporal Constraint Networks and Parameter Dependency Network, CORALS uses an agenda structure to manage conflicts and incidentally agenda-based conflict detection and conflict resolution algorithms. Another difference between CORALS and ASPEN lies in the way the initial plans are generated, as CORALS uses as a union of a set of local plans, either generated internally or provided by a separate planner. Also, CORALS handles uncertainty about the effects of actions. With regard to applications, ASPEN was mainly experimented with space mission planning problems, whereas CORALS is primarily for the CPM domain.

#### 4.4 Decision criteria

The validity criterion for a plan is the absence of conflicts, that is, none of its actions violates constraints. A suboptimal, but conflict-free, plan will be accepted as a solution, if there is not enough to achieve optimality. The latter is conveyed by an objective function referred to as the *plan-quality metric function* or (*plan-quality function* for short). This function takes a plan as input and returns a real number expressing the quality of the plan on a real-valued scale. The higher the value is, the better the plan is. The optimal plan is a conflict-free plan having the highest plan quality.

There can be different formulations of the notion of *plan quality* depending on the domain of applications and user preferences. Accordingly, in CORALS the plan-quality function is an option selected by the system user, *i.e.*, the naval warfare operator. For the CPM domain, the  $P_{SE}$  can be chosen as a quality criterion for the local plans and the  $P_{RA}$  for the global plan. The survival value of the targets offers an alternative plan quality criterion, which is also supported by CORALS. With this criterion, the CORALS user can attribute different values to the targets and compute a plan that minimizes the overall expected target survival value.

Knowing that CORALS proceeds by generating a sequence of conflict-free plans that are increasingly closer to the optimal, the operator can specify a maximum planning time, such that if it is reached, the best plan computed so far is returned, albeit not necessarily be optimal.

## 5. PLANNING ALGORITHMS

The algorithm `CORALSPANNER` described below is the entry point of `CORALS`. It takes as input a list of activities, a current state of the defending force, a goal, and domain knowledge and generates a global plan that assigns local plans to activities such that the set of all local plans is a conflict-free plan with a plan quality value. The global plan is guaranteed to be optimal only when the plan space is exhausted. The activities, state, goal, and domain knowledge are abstracted over to keep the details of the algorithms descriptions succinct.

### Algorithm 1 CORALS Planner Algorithm

```

1. Algorithm CORALSPANNER(Target activityslist[1...n])
2.   Plan plan ← ∅
3.   for i = 1 to n:
4.     Plan localplan ← LocalPlanner(activityslist[i])
5.     localplan ← LocalPlannerOptimizer(localplan)
6.     plan.add(localplan)
7.   return PlanMerger(plan)

```

*Main Entrance*

For each activity, a local plan is computed by the local planner (Line 4), optimized by a local plan optimizer (Line 5), and a global (multi-activity) plan is formed as the union of the optimal local plans (Line 6). Since the current global plan is a mere union of local plans, it likely contains conflicts. Thus the last step is to detect and resolve those conflicts; *i.e.*, to repair the plan (Line 7).

#### 5.1 Local plan optimizer

By definition, a local plan is optimal, with respect to  $P_{SC}$ , if and only if it there exists no other plan which has a higher  $P_{SC}$  value. The optimization of local plans is performed by invoking the algorithm `CORALSLOCALPLANOPTIMIZER`. The latter takes as input a local plan for an activity (target) and a specification of the  $P_{SC}$  functions for each action type (associated to a combat resource). Since the probability of success of an action highly depends on its start time, the local optimizer reschedules actions by using optimization techniques to find the peak of the  $P_{SC}$  function of each action of the local plan. The algorithm `CORALSLOCALPLANOPTIMIZER` returns an optimal local plan which corresponds to a plan where all actions are rescheduled to its optimal start time.


#### 5.2 Plan Merger

The `PLANMERGER` algorithm (Algorithm 2) is the most important and elaborate algorithm in `CORALS`. It takes as input a set of optimized local plans and returns a global conflict-free plan by merging local plans and repairing the resulting plan. The returned plan is optimal, if sufficient computation time and memory are available.

The plan merger process performs a best-first search to modify the input plan by removing conflicts. To this end, the algorithm generates a search graph representing the space of all possible plan modifications. More specifically, a node in the graph corresponds to a plan, and transition from plan  $p_1$  to plan  $p_2$  represents a transformation of  $p_1$  into a new plan  $p_2$ ; that is, a transformation that tries to

remove a conflict from the plan  $p_1$ . Thus a path in the graph is a sequence of plan modifications. The root plan is the initial global plan formed by the union of optimal local plans.

#### Algorithm 2 Plan Merger Algorithm



```

1. Algorithm PLANMERGER(Plan plan)
2.   Variable open, close, plan, newPlan, bestPlan
3.   open.add(plan)
4.   Repeat until HaltCondition(bestPlan, open)
5.     plan ← removeFirst(open)
6.     plan.conflicts ← DetectConflicts(plan)
7.     close.add(plan)
8.     if plan.conflicts.isEmpty() and plan.quality() > bestPlan.quality()
9.       bestPlan ← newPlan
10.    successors ← GenerateSuccessors(plan)
11.    for each newplan in successors
12.      newplan.hvalue ← HeuristicValue(plan)
13.      if (not open.findEqual(newPlan)) and (not close.findEqual(newPlan))
14.        open.add(newplan)
15.  return bestPlan

```

The search graph is generated and explored on the fly and is partitioned into two mutually exclusive lists: *open list* and *close list*. The open list contains the frontier plans, *i.e.*, plans that have not been expanded yet. The close list contains plans that have already been expanded, and hence should not be expanded any more should they be met on a different search path. The close list is required in order to avoid infinite cycling.

At the beginning, the input plan is added to the open list (Line 3). Conflicts are detected using the function *DetectConflicts* and the list of conflicting actions is attached to the plan (Line 6). The function *HaltCondition* returns true if the halt condition is met, at which point the planner stops and returns the best plan found so far. The halt condition is reached when the maximum available planning time has elapsed or the open list is empty. The latter case means that the search space has been entirely explored and returned plan is constructively proven to be optimal. The proof trivially follows from the exhaustion of the search space. As long as the halt condition is not met, the plan from the top of the open list is removed (Line 5), its conflicts are computed and attached to it (Line 6), and then it is added to the close list (Line 7), meaning that it has already been visited. A plan on the close list is not visited again (Lines 13). CORALS keeps track of the best plan found so far at Lines 8 and 9. The test on Line 8 ensures that only conflict-free plans are considered in keeping track of the best plan.

### 5.3 Conflicts Detection and Conflicts Resolution

The successors of the current plan in the search graph are computed using the function *GenerateSuccessor* (Line 10). This function takes as input a plan and a corresponding set of conflicting actions, and returns the set of plans obtained from the input by selecting one conflict applying a modification to try to resolve it. For each detected conflict, successors are generated that account for the different ways

to resolve the conflict. For example, successors are generated by rescheduling one of the actions at a different time or by completely removing it. Each modification gives rise to a single plan successor. The conflict in the new plan may not be completely resolved in a single transition of the search graph and may take a sequence of transitions to resolve a conflict; *i.e.*, more transitions in the search graph.

Prior to their resolution, conflicts are detected by the DETECTCONFLICTS algorithm (Algorithm 3). For the most part, constraints in the CPM planning domain are naturally specified as constraints on combat resources. Consequently, CORALS detects conflicts in terms of violation of the resource capacity constraints. More precisely, actions are modeled in such a way that their effects involve operations of using combat resources of finite capacities. Therefore a conflict occurs if and only if the availability of a resource is less than 0. The actions involved in the utilization of that resource are then known to be in conflict. Each plan in the search graph has an *agenda* attached to it. The agenda is a list of events sorted chronologically to keep track of the reservation and release of combat resources by actions. The capacity of a resource changes over time in accordance with its reservations and releases by actions. A conflict is declared on an action if the number of reservations for a resource exceeds its capacity for a specific time interval (Lines 25-26).

### Algorithm 3 Conflicts Detector Algorithm

```

1. Structure Reservation:
2.   Integer quantity
3. Structure RenewableResource:
4.   Integer capacity, quantity
5.   Reservation reservations[Action]
6. Structure Event:
7.   Action action
8.   Resource resource
9.   Integer time, quantity
10.
13. Algorithm DETECTCONFLICT(Plan plan, Resource[] res)
14.   Variable: events[time index], conflicts=∅
15.   for each Action a (in) plan :
16.     actionevents = a.getResourceEvents()
17.     for each Event e (in) actionevents :
18.       events[e.time].add(e)
19.   for t = 0 to maxtime :
20.     for each e in events[t] :
21.       ✓ e.resource.quantity += e.quantity (adding up event resource usage to resource itself)
21.       ✓ e.resource[e.action].quantity += e.quantity (adding up the reservation of related resource)
22.       for each r in res :
23.         if (r.quantity > r.capacity)
24.           for each a in plan:
25.             if r.reservations[a].quantity > 0
26.               conflicts.add(a)
27.       return conflicts
28.   return ∅

```

#### 5.4 Heuristics

CORALS uses heuristics to explore the most promising plans first by keeping plans in the open list sorted, so to explore them based on their respective heuristic value (*hvalue*). This value is computed by the heuristic function *HeuristicValue()* (Line 12) in Algorithm 2. The value is an estimation of how promising the plan is on a path to the optimal solution, which allows the most promising plans to be expanded first. The efficiency of the search process is directly influenced by the quality of the heuristic.

The heuristic value for a plan is related to the plan quality criterion. To illustrate, when  $P_{SC}$  is used as the plan quality criterion, the heuristic value of a plan is computed as the *maximal conflict-free subset of the plan*. This is an estimate of the lower bound on  $P_{SC}$  of best plan that could be derived from the current plan by resolving its conflicts. The *maximal conflict-free subset of a plan* means a conflict-free subset of the plan such that, adding any action (from the total plan) to the subset would result in a conflict. Here maximal refers to the fact that the subset is the largest (maximal cardinality) subset with non-conflicting actions. This does not refer to any optimality of the subset in terms of  $P_{SC}$ .

In fact, given a plan, there could be several different maximal subsets. An optimal maximal conflict-free subset is the one, among the possible subsets, that has the highest  $P_{SC}$ . The worst case complexity for computing an optimal maximal conflict-free subset of a plan is exponential in the number of actions in the plan. Therefore, the optimal maximal conflict-free subset can be used as the basis for lower-bound  $P_{SC}$  estimates, since the overhead resulting from the computation of the subsets would outweigh the savings obtained by exploring a smaller search graph. As a trade-off, maximal conflict-free subsets that are not necessarily proven optimal are computed using the *Maximal-Subset  $P_{SC}$  Lower Bound Estimate*.

Given a plan  $P$ , a maximal subset is computed by iteratively eliminating one action from  $P$ , until the current remaining subset of  $P$  is conflict-free. At every step, the action with the (estimated) minimal impact on the plan  $P_{SC}$  is removed. The time complexity of the *Maximal-Subset Lower Bound Estimate* method is  $O(n + c \cdot t)$ , where  $n$  is the number of actions in the plan,  $c$  is the number of conflicts and  $t$  is the number of different key time points. A key time point is defined as a time point where a combat resource is engaged or released. If two different combat resources are engaged at the same time, then this account for as just one time point (similarly for the release). Note that an action may involve several combat resources. Also, the resources can be engaged or released at the same time. Hence  $t$  is not necessarily twice the number of actions; it may be smaller or greater.

The Maximal-Subset Lower Bound Estimate heuristic is combined with an Enforced-Hill Climbing search strategy, which is a mixture of a global search strategy and the commonly used hill-climbing search strategy for local search methods [Hoffmann and Nebel 2001]. The idea is to find a conflict-free plan by using a hill-climbing strategy and then improve it by switching to a best-first strategy. Technically, at some point search is done by focusing on the best node among the successors of the current one (*i.e.*, hill-climbing) and later we switch to the best node among those not expanded yet (*i.e.*, a best-first). The depth up to which Enforced-Hill Climbing



is performed is controlled by an empirical parameter.

### 5.5 Execution Monitoring and Re-planning

The CPM domain is dynamic by its very nature. The tactical situation can change drastically during the plan elaboration and/or execution. New targets can appear, while others may disappear. Combat systems and the underlying resources can have failures too. Therefore the execution of a plan is monitored to detect changes that are not accounted for during planning (*i.e.*, contingencies). New plans are therefore generated to take into account changes in the tactical situation or in the status of combat systems.

## 6. PERFORMANCE EVALUATION

The performance evaluation of CORALS is presented here only from an algorithmic perspective. The reason behind this is threefold: (i) the primary focus of the current paper is only on the algorithmic aspects of CORALS; (ii) the planning and conduct of Human Factors experimentations require significant time and budget; and (iii) the results of such experimentations, once conducted, are often (if not always) sensitive and not publishable in the open literature. Nonetheless, we have been able to conduct a small amount of experimentation and to obtain a very positive feedback from naval warfare subject matter experts, who are currently using a modified version of CORALS as a decision support tool on a regular basis.

To illustrate the performance of CORALS, the Probability of Raid Annihilation  $P_{RA}$  plan quality is considered as a function of the planning time and the planning problem complexity in the CPM domain. One factor influencing the complexity of a CPM planning problem is the number of targets. From a human factor perspective, more targets lead to less reaction time and an increased cognitive complexity in the decisions to be made [Bos et al. 2005]. From an algorithmic perspective, the number of targets directly has a significant on the number of actions in each local plan, and indirectly on the number of actions in a global plan. An increase in the number of actions in the plan will increase the plan merging process, as it has to consider more options in order find a global plan. Furthermore, more actions will likely lead to more conflicts to detect and to resolve. This will contribute to further increase the complexity of the planning process.

The behavior and the spatial and temporal distribution of the targets in the tactical situation are additional important factors that may influence the planning complexity. In particular, targets arriving in quick succession can cause more potential conflicts among the actions. The spatial and temporal distributions of targets are conveyed by the target kinematics. The number, constraints, and characteristics of the involved combat resources also have a direct influence on the number of potential conflicts in plans.

The above factors are seen as dimensions in the “planning-complexity space”. Different types of scenarios can be defined by abstracting over a number of dimensions, such as focusing only on the number of targets, or on the number of targets and the distance that separates them. Given that the evaluation considers a subset of the complexity factors, the sample scenarios need to be statistically significant with respect to the factors that are abstracted away. To that end, for each complexity level, an empirically fixed number of samples (100, here), which



are instances of the scenario type at that level of complexity, by randomly varying the abstracted away factors. The reported performance metrics, for the selected level of complexity, is the average performance over all the samples. By choosing a large number of samples, the performance evaluation becomes more statistically significant at the cost of more time for conducting the experiment.

### 6.1 Scenarios generation

The generation of the scenarios for each experiment is automated using ad-hoc routines. For instance, in the case of scenarios where the number of targets is fixed, a routine was used to create the predefined number of targets, by randomly choosing their type from a set of eight types, their speed (from 200 to 1200m/s), their initial range (from 30 to 70km) and their bearing (between 0 and 360 degrees). The configuration of the defensive combat resources remains fixed throughout the experiments. It corresponds to a fictive warship of frigate type: one missile launcher; two fire control radars; two different guns (IRG; CIWS); four decoys (Chaff) launchers; two electronic measures (jammers). All experiments were run on an Intel(R) Core(TM) 2 Quad CPU Q6600 with 2GB of RAM. The operating system was Ubuntu Linux 8.10, and the Java Virtual Machine Version is 1.6.0.

### 6.2 Results

Figure 5 (a) shows the plan quality ( $P_{RA}$ ) as a function of the number of targets, where the planning time is fixed to 10 seconds. The MBLE heuristic is used, along with the Enforced-Hill Climbing, for which results correspond to a depth of 25. As may be expected, the plan quality decreases with the increasing number of targets. The decrease is not just because of the increase of the planning problem complexity, but also, and most importantly, because the number of available defensive combat resources and their configurations remain fixed, for more targets to defend against.

In Figure 5 (b), the number of targets is fixed to 10 and  $P_{RA}$  is shown as a function of the planning time. After 6.5 seconds, the plan quality does not improve much due mainly to the limited number of defensive combat resources. Although CORALS has not yet finished exploring the search space, to mathematically guarantee optimality, the efficiency of the heuristics used by CORALS allows the planner to converge to a high quality (given the resource constraints) plan very quickly. For the presented results, CORALS reaches the optimal plan around 10 seconds, but this can't be proven until the search space is exhausted.

Figure 5 (c) shows the evolution of the plan quality with the detection range of the targets. The latter have a fixed number (10) and are assumed to travel at different, but constant, speeds. The planning time is limited to 10 seconds. It is clear that the plan quality improves as the targets are more dispersed. The reason is that target dispersion gives CORALS the ability to re-allocate the same combat resource to different targets (at different times), thereby increasing  $P_{RA}$ .

All the result presented so far are based on the MLBE heuristic combined to Enforced-Hill Climbing heuristic (with depth=25). The benefits of the addition of Enforced-Hill Climbing are clearly shown in Figure 6. The figure shows the plan quality as a function of the number of targets, with planning time fixed to 10 seconds. The different curves correspond to different values of the depth used by the Enforced-Hill Climbing heuristic. A depth of 0 corresponds to no hill-climbing

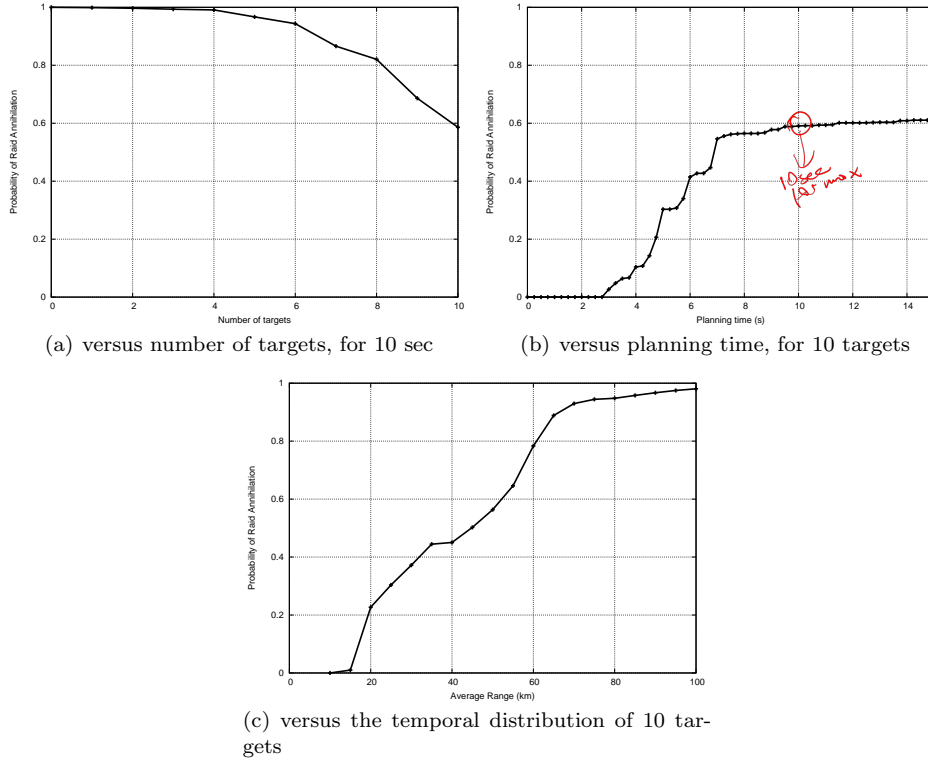


Fig. 5. Plan quality: Probability of Raid Annihilation

component in the search strategy, that is, MLBE alone. As previously noted, the plan quality decreases with the increasing number of targets, independently of the depth value. However, the decrease is slowed as the depth value is increased.

Figure 7 shows the plan quality as a function of the planning time, with the number of targets fixed to 10. The results show clearly the positive impact of higher hill-climbing depths on the plan quality.

## 7. CONCLUSION

This paper presented CORALS, an automated planning system developed to support the command team of a defending force in optimally planning actions against a set of threats. CORALS proceeds by generating a set of individual-target plans, and then merges them by detecting and resolving conflicts using a plan repair strategy. This approach is also adequate for revising a previously generated plan to account for dynamic changes in the tactical situation. CORALS can handle domains characterized by durative actions, concurrent actions, probabilistic effects, and goal deadlines, where the goal is conveyed by an objective function to optimize. The CORALS planner has been evaluated on realistic naval warfare CPM scenarios and has showed very conclusive performance. A classified version of the planner has also been tested live in operational environment, during a naval warfare exercise.

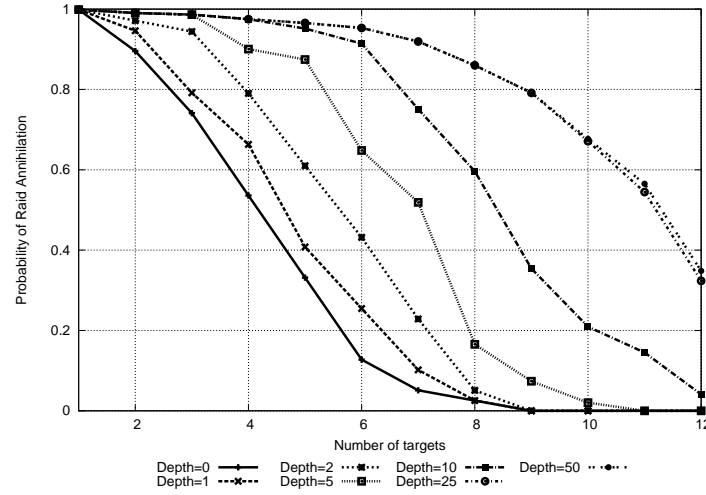


Fig. 6. Combined MLBE and Enforced-Hill Climbing heuristics on a varying number of targets

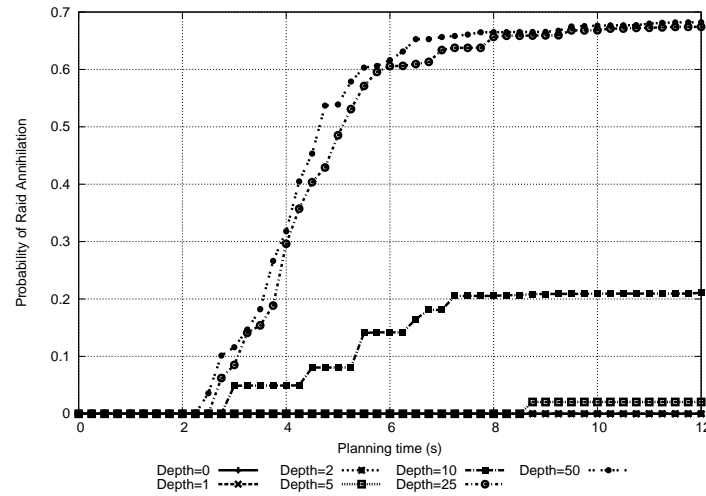


Fig. 7. Enforced hill-climbing performance for 10 targets

The current design of CORALS follows a centralized planning approach. A work in progress intends to produce, using a decentralized planning approach, conflict-free and optimized response plans, taking into account the geographical dispersion of both the combat resources, the protected assets, the decision-makers (planners). Such a problem fits naturally within the framework of distributed problem solving and, more precisely, multi-agent planning. Also, in domains as critical as CPM, it may be interesting to make the plan generation process a cooperative task between the automated planner and the human operator. Such a mixed-initiative planning setting is being explored, to provide the operator with different options

for suggesting or enforcing planning steps and constraints.

#### ACKNOWLEDGMENTS

We wish to thank Taek-Sueng Jang for his valuable contribution to the implementation of the presented planner.

#### REFERENCES

- AHUJA, K., KUMAR, A., JHA, K., AND ORLIN, J. 2003. Exact and heuristic algorithms for the weapon target assignment problem. *MIT Sloan Working Paper No. 4464-03*.
- BENASKEUR, A., KABANZA, F., BEAUDRY, E., AND BEAUDOIN, M. 2008. A probabilistic planner for the combat power management problem. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS)*. 12–19.
- BERTSEKAS, D., HOMER, M., LOGAN, D., PATEK, S., AND SANDELL, N. 1999. Missile defence and interceptor allocation by neuro-dynamic programming. *IEEE Transactions on Systems, Man and Cybernetics, Part A* 30, 1 (January), 42–51.
- BOHACHEVSKY, I. AND JOHNSON, M. 1993. Optimal deployment of missile interceptors. *American Journal of Mathematical and Management Sciences* 13, 1-2, 53–82.
- BOS, J., REHAK, L., KEEBLE, A., AND LAMOUREAUX, T. 2005. Tactical planning and response management: Investigating a cognitive work analysis approach to the development of support concepts. Tech. rep., DRDC Atlantic.
- BRUCKER, P. AND KNUST, S. 2006. *Complex Scheduling*. Springer.
- BUFFET, O. AND ABERDEEN, D. 2006. The factored policy gradient planner. In *Fifth International Planning Competition*.
- ESTLIN, F. F. T., MUTZ, D., AND CHIEN, S. 1999. Artificial intelligence planning to generate antenna tracking plans. In *Proc. of Innovative Applications of Artificial Intelligence (IAAI)*. 856–863.
- FOX, M. AND LONG, D. 2003. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20, 61–124.
- HADJ-ALOUEANE, A., BEAN, J., AND MURTY, K. 1999. A hybrid genetic/optimization algorithm for a task allocation problem. *Journal of Scheduling* 2, 189–201.
- HOFFMANN, J. AND NEBEL, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14, 253–302.
- HOSAIN, P. AND ATHANS, M. 1990. Some analytical results for the dynamic weapon-target allocation problem. *Naval Research Logistics Journal*.
- LLOYD, S. AND WITSENHAUSEN, H. 1986. Weapons allocation is np-complete. In *IEEE Summer Conference on Simulation*.
- MATLIN, S. 1970. A review of the literature of the missile-allocation problem. In *Operations Research*. 334–373.
- MAUSAM AND WELD, D. S. 2006. Probabilistic temporal planning with uncertain durations. In *National Conference on Artificial Intelligence (AAAI)*.
- MEULEAU, N., HAUSKRECHT, M., KIM, K.-E., PESHKIN, L., KAEHLING, L., DEAN, T., AND BOUTILIER, C. 1998. 15<sup>th</sup> national conference on artificial intelligence (AAAI). In *Operations Research*. 165–172.
- OSBORNE, D. 1993. Ship self-defence against air threats. *Johns Hopkins APL Technical Digest*. 12(2):125-140..
- YOUNES, H. AND SIMMONS, R. 2004. Policy generation for continuous-time stochastic domains with concurrency. In *International Conference on Automated Planning and Scheduling (ICAPS)*. 325–334.

Submitted March 2010; Revised June 2010; Accepted July 2010