Cooperative Unmanned Air Vehicles

A. Girard et al.

Michigan/AFRL Collaborative Center in Control Sciences
University of Michigan
Ann Arbor, Michigan

September 23, 2009

Dynamic Observations in Discrete Event Systems

W. Wang & A. R. Girard

Michigan/AFRL Collaborative Center in Control Sciences
University of Michigan
Ann Arbor, Michigan

September 23, 2009

Contents

- Introduction
- Dynamic Observations
- Verification of Codiagnosability under Dynamic Observations
- The Transformation of Coobservability to Codiagnosability
- Optimizing Sensor Activations
- Conclusion and Future Work

Motivation

- Unmanned Aircraft Systems (UAS) & Discrete Event Systems (DES)
  - Supervisory control is indispensable for automation in UAS
  - Precedence order of event occurrences is a key factor
- Observations are dynamic in military operations
  - Dynamic observations are caused by agents turning their sensors on/off
  - Agents occasionally communicate their observations
  - The presence of a reconnaissance UAV when an event occurs allows the UAV to sense that event

Figure: Draganflyer X6

Figure: Wireless Sensor

System Model

- The dynamics of DES are driven by event occurrences
- E is the set of events
- The set of all finite strings of events in E is denoted by E^*
- A *language* L defined on E is a subset of E^* : $L \subseteq E^*$
- The behavior of a DES is described as a *language*
- A language can be modeled by an automaton (finite or infinite state)
- The *prefix-closure* of L is: $\overline{L} := \{s \in E^* : (\exists t \in E^*) st \in L\}$
- A language L is prefix closed iff $L = \overline{L}$

System Model

- Finite State Automaton $G = (X, E, \delta, x_0)$ where
  - X is the state space with finite cardinality
  - E is the set of events
  - $\delta : X \times E \to X$ is the transition function
  - x_0 is the initial state
- The operation of G always starts at the initial state x_0
- δ is extended recursively to the domain $X \times E^*$ in the following manner: $\delta(x_0, \epsilon) = x$; for $s \in E^*$ and $e \in E$, $\delta(x_0, se) = \delta(\delta(x_0, s), e)$
- The language generated by G is $\mathcal{L}(G) = \{s \in E^* : \delta(x_0, s) \text{ is defined}\}$

System Model

- The set of diagnostic or control agents is $\mathcal{A}$
- When the local behavior is modeled by automaton G_i for each $i \in \mathcal{A}$, the global behavior can be synchronized by using the *parallel composition* technique
  - In parallel composition, a common event (an event in $E_i \cap E_j$) can only be executed if the two automata both execute it simultaneously
  - Other events can be executed whenever possible

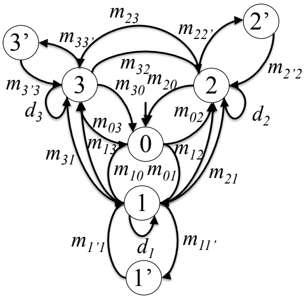
Synchronization of Local Models: 40 by 40 Mile Problem

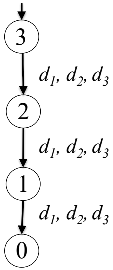
To model a distributed system globally, we may start with building local models. Then, synchronize local models by parallel composition for global model. We consider the 40 by 40 mile problem for example

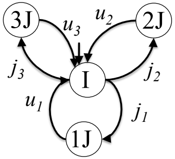
- We have a bomber with three bombs and a jammer. Hostile side has three radars
- Our goal is to eliminate hostile radars
- Our bomber cannot go directly into a radar detection region if the radar is alive and not jammed
- When a radar is jammed, its detection region shrinks and then our bomber can go close enough to disable it

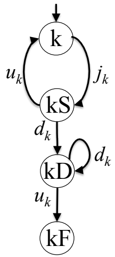
Synchronization of Local Models: 40 by 40 Mile Problem

- Event labels: m_{kl} is for bomber moving from k to l, d_k is for disable radar k, j_k is for jamming radar k, and u_k is for unjamming radar k
- Figures are models for bomber mobility, bomber attack, jammer, and one of three radars, respectively

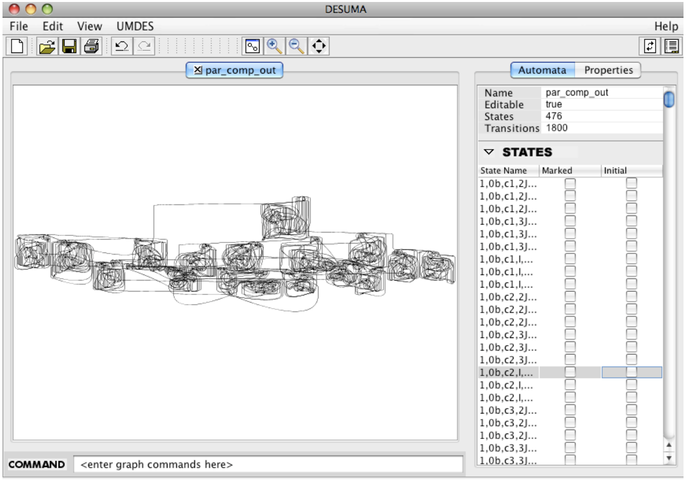








Synchronization of Local Models: 40 by 40 Mile Problem



Name	par_comp_out
Editable	true
States	476
Transitions	1800

State Name	Marked	Initial
1,0b,c1,2J...	☐	☐
1,0b,c1,2J...	☐	☐
1,0b,c1,2J...	☐	☐
1,0b,c1,3J...	☐	☐
1,0b,c1,3J...	☐	☐
1,0b,c1,3J...	☐	☐
1,0b,c1,I,...	☐	☐
1,0b,c1,I,...	☐	☐
1,0b,c1,I,...	☐	☐
1,0b,c2,2J...	☐	☐
1,0b,c2,2J...	☐	☐
1,0b,c2,2J...	☐	☐
1,0b,c2,3J...	☐	☐
1,0b,c2,3J...	☐	☐
1,0b,c2,3J...	☐	☐
1,0b,c2,I,...	☐	☐
1,0b,c2,I,...	☐	☐
1,0b,c2,I,...	☐	☐
1,0b,c3,2J...	☐	☐
1,0b,c3,2J...	☐	☐
1,0b,c3,2J...	☐	☐
1,0b,c3,3J...	☐	☐
1,0b,c3,3J...	☐	☐

Figure: The global model for the 40 by 40 mile problem

System Model

- Not all events are observable to every agent. For each agent $i \in \mathcal{A}$, we define $E = E_{o,i} \dot{\cup} E_{uo,i}$ where
  - $E_{o,i}$: potentially observable event set to agent $i \in \mathcal{A}$
  - $E_{uo,i}$: unobservable event set to agent $i \in \mathcal{A}$
- Let $E_o = \cup_{i \in \mathcal{A}} E_{o,i}$ and $E_{uo} = E \setminus E_o$
- An automaton that has no deadlocked state (a state that has no outgoing event) is called live

System Model

We consider a military operation scenario as follows.

- Suppose that we know there is a hidden hostile installation, comprised of a missile launcher and a radar, in a region r, but we don't know the precise position of the installation
- Suppose we have a reconnaissance unmanned aerial vehicle (UAV), an attacker, and a jammer
- The reconnaissance UAV is able to enter or leave the region r
- The goal of the hostile side is to launch a missile to attack us, whereas our goal is to eliminate the hostile missile launcher

System Model

The corresponding event set describes the example:
$E = \{e, l, jr, uj, dm, sl, lr, or, to, rr, lm\}$, where

- e: reconnaissance UAV enters region r
- l: reconnaissance UAV leaves region r
- jr: jammer jams hostile radar
- uj: jammer unjams hostile radar
- dm: attacker disables hostile radar
- sl: hostile side starts to set missile launcher
- lr: hostile side launcher becomes ready
- or: hostile side opens launcher radar
- to: a time threshold has passed since hostile side opened radar
- rr: hostile side radar becomes ready, and
- lm: hostile side launches a missile

System Model

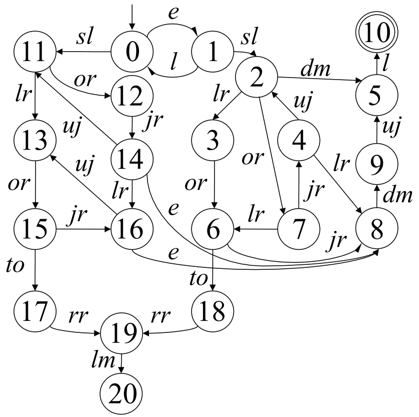


Figure: A Military Operation can be modeled by DES

System Model

- Suppose that the set of controllable events is $E_c = \{e, l, jr, uj, dm, to\}$ and the set of uncontrollable events is $E_{uc} = \{sl, lr, or, rr, lm\}$
- For the hostile side to successfully launch a missile, it needs to complete all events (tasks) sl, lr, or, rr, and lm
- Events must not necessarily occur in this order, but event sl must occur before all other events, or must occur before rr, and lm must occur after all other events
- Consequently, to prevent the hostile side from launching a missile, we either disable the launcher (enabling dm) or jam the radar (enabling event jr) before lm occurs

System Model

- However, since rr and lm are uncontrollable, we need to disable to at states 6 and 15 at the latest, if we want to prevent event lm from occuring at state 19
- Since the missile launcher is hidden, we are only able to disable it after the hostile side has started to set the launcher (after an occurrence of sl)
- We assume the hostile side needs to reopen its radar after it's been jammed and then unjammed
- Our goal is achieved when the system ends up at state 10, which is marked by double circles

System Observations

- In this scenario, observations of events are *dynamic*
- Since the reconnaissance UAV is not in region r when the system is at state 0, we cannot observe the occurrence of sl at state 0
- Therefore, we are not able to evaluate how long the hostile side prepares its launcher and are not able to observe the occurrence event lr at states 11, and 14
- On the other side, since the reconnaissance UAV is in the region r when the system is at state 1, we are able to observe the occurrence of sl at state 1

System Observations

- Thus, we are able to evaluate how long the hostile side prepares its launcher and able to observe the event lr at states 2, 4, and 7
- The observations for events or and to are also dynamic. An occurrence of event or is only observed when the corresponding sensor for detecting a radar signal is activated,
- and whether or not an occurrence of event to is observed depends on the observation of the occurrence of event or at the closest upstream

System Observations

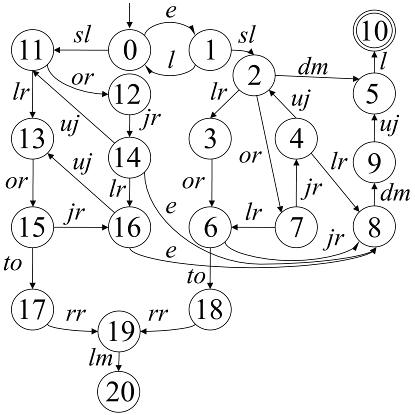


Figure: Observations of sl, lr, or, and to depend on system dynamics

Contents

- Introduction
- Dynamic Observations
- Verification of Codiagnosability under Dynamic Observations
- The Transformation of Coobservability to Codiagnosability
- Optimizing Sensor Activations
- Conclusion and Future Work

Observation Mapping

- *Dynamic observations:* the agents' observations depend on both of
  - which event is observed and
  - which trajectory of the system dynamics is followed
- Whether or not an event occurrence is observable by agent i, $i \in \mathcal{A}$, is described by the *observation mapping*

$$\omega_i : \mathcal{L}(G) \to 2^{E_o}$$

- For a trajectory $s \in \mathcal{L}(G)$, $\omega_i(s)$ is the subset of observable events E_o after s.

Information Mapping

- The information mapping (or projection) $\theta_i : \mathcal{L}(G) \rightarrow E_o^*$ is:
    - for the empty string ϵ, $\theta_i(\epsilon) = \epsilon$, and
    - for all $s, se \in \mathcal{L}(G)$ with $e \in E$,

$$\theta_i(se) = \begin{cases} \theta_i(s)e & \text{if } e \in \omega_i(s) \\ \theta_i(s) & \text{otherwise} \end{cases}$$

- After the occurrence of s, the next event e is *seen* by agent i when it occurs after s if and only if it is in $\omega_i(s)$

Event Diagnosis

- Notations for event diagnosis
  - The set of fault events to be diagnosed is denoted by $E_f \subseteq E$
  - The set of fault events is partitioned into different fault types: $E_f = E_{f_1} \dot{\cup} \ldots \dot{\cup} E_{f_K}$
  - $s \in \Psi(E_{f_k})$ means that the last event of a trace $s \in \mathcal{L}(G)$ is a fault event of type f_k
  - $\mathcal{L}(G)/s$ denotes the postlanguage of $\mathcal{L}(G)$ after s
  - $E_{f_k} \in s$ denotes that $PC(s) \cap \Psi(E_{f_k}) \neq \emptyset$
  - $|s|$ is the number of event occurrences in s
- The *objective* is to identify the occurrence of events, if any, in the set of fault events by tracking the observed traces

Event Diagnosis

Definition

A prefix-closed and live language $\mathcal{L}(G)$ is said to be codiagnosable with respect to θ_i, $i \in \mathcal{A}$, and Π_f on E_f if the following holds:
$(\forall\, k \in \Pi_f)(\exists\, n_k \in \mathbb{N})(\forall\, s \in \Psi(E_{f_k}))(\forall\, t \in \mathcal{L}(G)/s)[|t| \geq n_k \Rightarrow CD]$
where the codiagnosability condition CD is
$(\exists\, i \in \mathcal{A})(\forall\, \mu \in \mathcal{L}(G))\theta_i(\mu) = \theta_i(st) \Rightarrow E_{f_k} \in \mu$

A system is said to be codiagnosable if any trace that contains a type k fault event can be distinguished by at least one agent from all traces without a type k fault event within finite delay

Coobservability for Control

- Control is necessary because the uncontrolled $DES\ G$ may violate *safety* or *nonblocking* specifications
  - The specifications on safety and nonblocking are referred to as "legal behavior" in DES
- The legal behavior K is described as a subset of $\mathcal{L}(G)$
- The goal of decentralized supervisory control is to find local supervisors such that the supervised system, denoted by $\wedge_{i=1}^{N} S_i/G$, generates the legal language K, that is, $\mathcal{L}(\wedge_{i=1}^{N} S_i/G) = K$

Coobservability for Control

- The notion of coobservability is used to classify whether or not local controllers are able to make sufficient observations of the system such that the correct control decisions can be made
- Notations for coobservability:
  - $E_{c,i} \subseteq E$, $i \in \mathcal{A}$, is the set of events that are controllable to agent i
  - $E_c = \cup_{i \in \mathcal{A}} E_{c,i}$ is the set of controllable events,
  - ω_i, $i \in \mathcal{A}$, is the observation mapping defined on language K
  - For an event $e \in E_c$, $A^c(e)$ is the set of agents that are able to control e, i.e., $A^c(e) = \{i \in \mathcal{A} : e \in E_{c,i}\}$
  - Suppose $H = (X_H, E, \delta_H, x_0)$ is the automaton that generates language K

Coobservability for Control

Definition

A prefix-closed language $K = \mathcal{L}(H) \subseteq \mathcal{L}(G)$ is *coobservable* with respect to $\mathcal{L}(G)$, and $E_{c,i}$, $i \in \mathcal{A}$, if for all $s \in K$ and $e \in E_c$ with $se \in \mathcal{L}(G)$, the existence of $s_i e \in K$ with $\theta_i(s_i) = \theta_i(s)$ for all $i \in A^c(e)$ implies $se \in K$, where θ_i is the information mapping for supervisor i corresponding to ω_i.

- Intuitively, in a decentralized system G, the legal behavior K is said to be coobservable if for any controllable event $e \in E_c$ and trace $s \in K$, that, if followed by e, leads to an illegal trace in $\mathcal{L}(G)$, there is at least one agent that is able to control e and can distinguish the trace from all other traces t such that $te \in K$
- In this way, e can be disabled after s without affecting the legal behavior of the system

Contents

- Introduction
- Dynamic Observations
- Verification of Codiagnosability
- The Transformation of Coobservability to Codiagnosability
- Optimizing Sensor Activations
- Conclusion and Future Work

Transition-based Dynamic Observations

- The set of transitions of G is
  $TR(G) = \{(x,e) \in X \times E : \delta(x,e) \text{ is defined}\}$
- The set of transitions that can be observed by an agent i is specified by an index function $I_i : TR(G) \rightarrow \{0,1\}$
  - $I_i(x,e) = 1$ means that the transition (x,e) is observable to diagnosing agent i and
  - $I_i(x,e) = 0$ means that it is not
- The observation mapping ω_i corresponds to I_i as
  $(\forall e \in E_o)(\forall s \in \mathcal{L}(G)) e \in \omega_i(s) \Leftrightarrow I_i(\delta(x_0,s),e) = 1$

Transition-based Dynamic Observations

- For the construction of the verifier, the pseudo index function $I_p : TR(G) \rightarrow \{0, 1\}$ is defined by

$$I_p(x,e) = \begin{cases} 1 & (\exists\, i \in \mathcal{A})\ I_i(x,e) = 1; \\ 0 & \text{otherwise} \end{cases}$$

- A verifier updates its system estimation for an occurrence of event e when the corresponding $I_p(x,e) = 1$

The Definition of Cluster Automata

- Under the observation I_i of diagnosing agent $i \in \mathcal{A}$, *cluster* $c_i(x)$ is the subautomaton of the accessible, fault free part of G whose
  - initial state x is the initial state or is entered by some observable transition in G and
  - whose set of transitions is the maximum set of unobservable and reachable transitions from state x

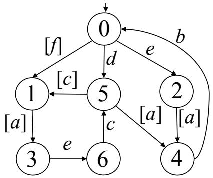


Figure: Observations for agent i

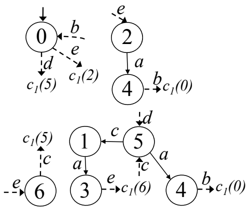


Figure: The set of corresponding clusters

The Transition Function

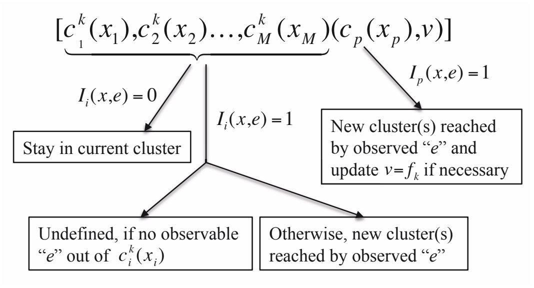

$[\underbrace{c_1^k(x_1), c_2^k(x_2) \dots, c_M^k(x_M)}(c_p(x_p), v)]$
$I_i(x,e)=0$
Stay in current cluster
$I_i(x,e)=1$
Undefined, if no observable "e" out of $c_i^k(x_i)$
Otherwise, new cluster(s) reached by observed "e"
$I_p(x,e)=1$
New cluster(s) reached by observed "e" and update $v=f_k$ if necessary

Figure: The result of a transition function is a set of vectors which illustrate the possible combination of clusters for each agent

The Transition Function

- $[c_1^k(x_1), \ldots, c_M^k(x_M), (c_p(x_p), v)]$ is a state in the verifier means that there is at least one trace $t \in \mathcal{L}(G)$ that ends up at $c_p(x_p)$ such that, for each $i \in \mathcal{A}$, there exists a type k faulty free trace $t_i \in \mathcal{L}(G)$ that ends up at state x_i with last event observable to i that looks the same as t
- Then, for all possible $tt'e \in \mathcal{L}(G)$ such that $t' \in E^*$ is unobservable to any agent and $e \in E$ is observable to some agents, one purpose of the transition function is to capture all possible vector of type k fault free traces $[t_1t'_1, \ldots, t_Mt'_m]$ such that $t_it'_i$ looks the same to $tt'e$ for agent $i \in \mathcal{A}$
- Instead of illustrating such vector of traces, this is done by illustrating corresponding vectors of clusters
- The illustration is based on the previous cluster and whether or not the occurrence of e is observed by the agent

The Transition Function

- Another purpose of transition function is to keep track of whether or not $tt'e$ contains a type k fault event
- Correspondingly, the Nf_k label at the part of the vector $(c_p(\delta(x, e)), v)$ for pseudo agent may change to f_k
- It depends on whether the unobservable subtraces t' of $tt'e$ contain a type k fault event or not

The Cluster-based Verifier

- Starting with $[c_1^k(x_0), \ldots, c_M^k(x_0), (c_p(x_0), Nf_k)]$, a cluster-based verifier can be constructed by recursively applying the transition function
- By the definition of codiagnosability, we have that G is not codiagnosable under observation $I_i, 1 = 1, \ldots, M$, iff
  1. a state $(x^D, 1) \in X^D$ is found with corresponding $c_p(x) \in \mathcal{C}_p^{f_k-cycle}$, or
  2. $c_p(x) \in \mathcal{C}_p^{cycle}$ with x^D labeled f_k, or if
  3. a cycle with f_k label exists in the C-VERIFIER

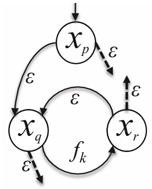


Figure: $c_p(x)$ in (1)

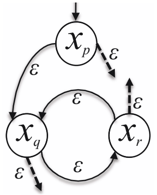


Figure: $c_p(x)$ in (2)

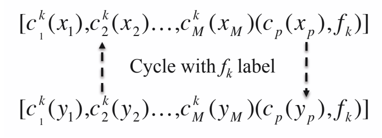

$[c_1^k(x_1), c_2^k(x_2)\ldots, c_M^k(x_M)(c_p(x_p), f_k)]$
Cycle with f_k label
$[c_1^k(y_1), c_2^k(y_2)\ldots, c_M^k(y_M)(c_p(y_p), f_k)]$

Figure: f_k labeled cycle in (3)

Examples of Cluster-based Verifier

- The following example shows a cluster-based verifier

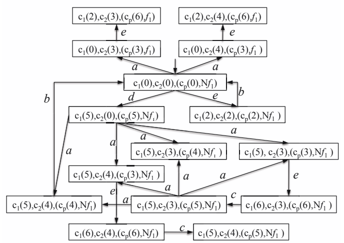


Figure: A cluster-based verifier: the system is codiagnosable

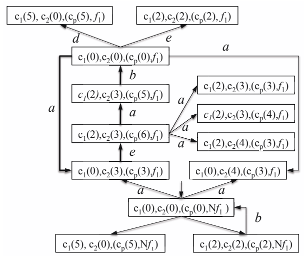


Figure: The system is no longer codiagnosable after changing observations

Complexity of the Cluster-based Verifier

Theorem

For a system modeled by automaton G with a fixed number of diagnostic agents, the verification of codiagnosability for transition-based dynamic observations is in worst-case of polynomial complexity in the size of the state space X of G

Summary

- We have presented a new procedure for the verification of the properties of diagnosability and codiagnosability in DES
- Our approach was specifically developed to handle the case of transition-based dynamic observations
- For a fixed number of agents, our verifier is in the worst-case of polynomial time in the state space of the system
- The new testing procedure that we have developed will be very useful in solving sensor activation problems or distributed diagnosis with communication problems

Contents

- Introduction
- Dynamic Observations
- Verification of Codiagnosability under Dynamic Observations
- The Transformation of Coobservability to Codiagnosability
- Optimizing Sensor Activations
- Conclusion and Future Work

Coobservability vs Codiagnosability

- The notion of codiagnosability is used to classify whether or not any traces that contain a fault event can be distinguished by at least one agent from all traces without the fault event within finite delay
- The notion of coobservability is used to classify whether or not local controllers are able to make sufficient observations of the system such that the correct control decisions can be made
- It was usually believed that the problem of coobservability was more complicated than the problem of codiagnosability
- However, we have shown that the problem of coobservability can be transformed to the problem of codiagnosability

The Transformation Algorithm

Algorithm COOBS-TO-CODIAG-I:

1. Set $\tilde{H}(e) \leftarrow H$ and add state d to state space of $\tilde{H}(e)$. Then, add self-loop (d, v) into $TR(\tilde{H}(e))$, i.e., $\delta_{\tilde{H}(e)}(d, v) = d$
2. For all $x \in X_H$, if $(x, e) \in TR(G) \setminus TR(H)$, add transition (x, f) to $TR(\tilde{H}(e))$ with $\delta_{\tilde{H}(e)}(x, f) = d$; if $(x, e) \in TR(H)$, add transition (x, u) to $TR(\tilde{H}(e))$ with $\delta_{\tilde{H}(e)}(x, u) = d$
3. We add an observable self-loop with event label $z \notin E$ at each state $x \in X_H \subseteq X$ that is a deadlock state in X
4. For all $i \in A^c(e)$, we specify observation mapping $\omega_{i,\tilde{H}(e)}$ for $\mathcal{L}(\tilde{H}(e))$ as follows. For all $s \in \mathcal{L}(\tilde{H}(e))$ such that $s \in \mathcal{L}(H)$ and the last event of s is not z, set $\omega_{i,\tilde{H}(e)}(s) \leftarrow \omega_i(s)$. For all $sz \in \mathcal{L}(\tilde{H}(e))$, set $\omega_{i,\tilde{H}(e)}(s) \leftarrow \{z\}$. For all $sfv^n, tuv^n \in \mathcal{L}(\tilde{H}(e))$ and $n \in \mathbb{N}$, set $\omega_{i,\tilde{H}(e)}(sfv^n) \leftarrow \{v\}$ and $\omega_{i,\tilde{H}(e)}(tuv^n) \leftarrow \{v\}$

The Transformation Algorithm

- In Algorithm COOBS-TO-CODIAG-I, we added a state d in the original system and a observable self-loop with event label v at state d
- The transformation is done for each controllable event $e \in E_c$ individually
- If at a state where e needs to be disabled, we connect that state to state d using unobservable event u
- If at a state where e cannot be disabled, we connect that state to state d using unobservable fault event f

The Transformation Algorithm

- In this way, for two arbitrary traces s and t, if the controller needs to disable e after s but cannot disable e after t, we have $\theta_i(s) = \theta_i(t)$ implies $\theta_i(sfv^n) = \theta_i(tuv^n)$ for all $n \in \mathbb{N}$
- That is, if two traces s and t cannot being distinguished by any agent and have a control conflict in the original system, they will cause a violation of codiagnosability in the transformed system
- Suppose agent i need to disable event b at state 2 but nowhere else

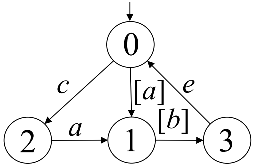


Figure: The legal behavior H and its observations to agent i.

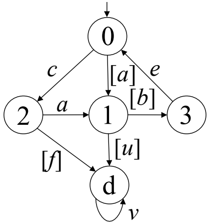


Figure: The automaton $\tilde{H}(b)$ and its observations to an agent i.

Summary

- We have presented an algorithm to transform the problem of coobservability to the problem of codiagnosability
- It is shown that, after the transformation, the problems of observability and coobservability are a special class of the problems of diagnosability and codiagnosability
- This enables us to leverage the large literature available for codiagnosability to solve problems of coobservability
- Because of their computational efficiency, these algorithms can be used for transforming large systems in practice

Related Publications

- Journal
    - W. Wang, A. R. Girard, S. Lafortune, and F. Lin "Codiagnosability and Coobservability in Dynamic Observations", revising for the resubmission to IEEE Transactions on Automatic Control, whole paper
- Conference
    - W. Wang, A. R. Girard, S. Lafortune, and F. Lin "The Verification of Codiagnosability in the Case of Dynamic Observations", in Proc. 2009 European Control Conference, Aug. 2009
    - W. Wang and A. R. Girard "Transformation of Coobservability to Codiagnosability in Dynamic Observations", submitted to 2010 American Control Conference
- Other
    - "The Verification of Co-Observability When Observations are Transition-Based Event Occurrences", in preparation

Contents

- Introduction
- Dynamic Observations
- Verification of Codiagnosability under Dynamic Observations
- The Transformation of Coobservability to Codiagnosability
- Optimizing Sensor Activations
- Conclusion and Future Work

Active Sensing of Partially Observed DES

- Estimation and sensor activation (SA) are *interdependent*!
  - What you have activated in the past affects your estimation (θ_i)
  - What you estimate influences your future activation decisions (ω_i)
- Consequently, the sensor activation policy (SAP) ω_i is a subclass of the observation mapping that satisfies feasibility for sensor activation
  - If two strings "look the same," then they must have same activation decision on a *common possible event*
  - Formally, ω is said to be *feasible* if $(\forall e \in E)(\forall se, s'e \in \mathcal{L}(G))\ \theta^{\omega}(s) = \theta^{\omega}(s') \Rightarrow [e \in \omega(s) \Leftrightarrow e \in \omega(s')]$

Active Sensing of Partially Observed DES

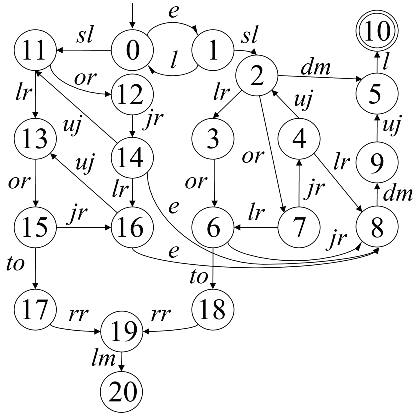


Figure: Whether or not an occurrence of event or is observable depends on whether our radar is turned on/off

Active Sensing of Partially Observed DES

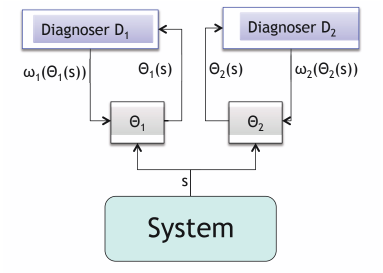


Figure: SA for diagnosis

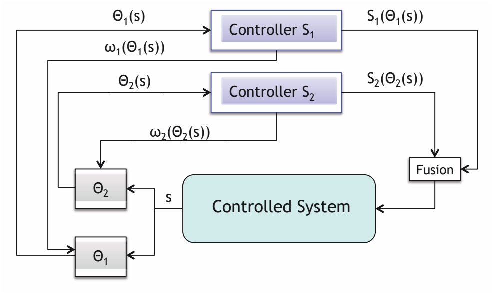
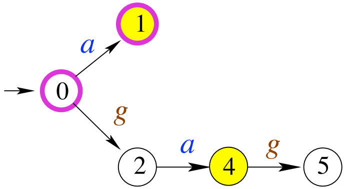


Figure: SA for control

Active Sensing of Partially Observed DES

Example of feasibility for SAP:

- If a is activated initially, then:
- Not activating g initially *and* a after string g is not *feasible*!
- after ϵ and g, we must have the same activation decision for a



Active Sensing of Partially Observed DES for Event Diagnosis

- Automaton G and set of agents $\mathcal{A}$
- Potentially Observable Event Set E_o of G.
- Set of fault events to be diagnosed $E_f = \{f\}$
- **When to activate sensors?**
  - Activate only if necessary, but enough to diagnose E_f

Language-based Partition

Why partition the language?

- Sensor Activation Policy: $\omega : \mathcal{L}(G) \rightarrow 2^{E_o}$
- Language $\mathcal{L}(G)$ typically has infinite cardinality
- Solution space of the sensor activation problem needs to be finite
- Restrict SAP to be a subset of a finite partition of $\mathcal{L}(G)$
    - All strings in the same element of partition have the same activation decision for their last event

Language-based Partition

- Formally, Δ is Language-based Partition (LBP) if its elements δ_j, $j = 0, 1, \ldots, m$ satisfy
  - Δ is a partition of $\mathcal{L}(G)$
  - $\delta_0 = \{\varepsilon\}$
  - Any strings in the same element of the partition Δ must have the same last event
- SAP: $\Omega \subseteq \Delta$
  - For each $\delta_i \in \Omega$, the last event in strings in δ_i is activated

Problem Statement

Given: G, E_o, E_f, and Δ
find: $\Omega^* \subseteq \Delta$ such that

- Ω^* satisfies the diagnosability requirement
- Ω^* satisfies the feasibility requirement
- Ω^* is a minimal set

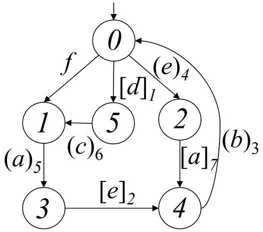


Figure: A Minimal Solution: ( ) activated; [ ] not activated

Main Theorems

Theorem

[Monotonicity]
Let Ω_1 and Ω_2 be two feasible SAP, such that $\Omega_1 \subseteq \Omega_2$. Then, the system is diagnosable for Ω_1 implies that it is diagnosable for Ω_2.

Theorem

[Existence of Maximum Element]
Let Ω be an SAP. Then there exists a **maximum feasible subpolicy** $\Omega^{\uparrow F}$ *that contains all $\Omega_F \subseteq \Omega$ that are feasible.*

Active Sensing: MIN-SEN-DIAG Algorithm

- Suppose that (i) $\Omega \subseteq \Delta$ is feasible, (ii) the system is diagnosable for Ω, (iii) $D \subseteq \Delta$ is the set whose elements have been examined but failed to be removed from Ω
- For a $\delta_i \in \Omega \setminus D$, let $\Omega_{test} = \Omega \setminus \{\delta_i\}$ and, then, calculate $\Omega_{test}^{\uparrow F}$
  - If there exists $\delta_j \notin \Omega_{test}^{\uparrow F}$ such that $\delta_j \in D$, or if the system is not diagnosable for $\Omega_{test}^{\uparrow F}$, then the system is not diagnosable for any feasible subsets of Ω_{test}
    $\Rightarrow$ Keep δ_i activated and set $D \leftarrow D \cup \{\delta_i\}$
  - If the system is diagnosable for $\Omega_{test}^{\uparrow F}$, then δ_i need not be activated
    $\Rightarrow$ Reinitialize Ω to $\Omega_{test}^{\uparrow F}$
- Proceed until $\Omega = D$. Then, set $\Omega^* \leftarrow \Omega$. A minimal (feasible and diagnosable) solution is found

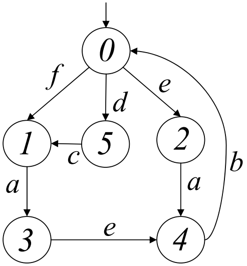
Active Sensing: $\uparrow F$ and Window Partitions

The set $\Delta^w = \{\delta_i \subseteq \mathcal{L}(G) : i = 0, \ldots, m\}$ is called an n-Window-Partition of $\mathcal{L}(G)$ if

- For all $u \in \mathcal{L}(G)$, if $\|u\| < n$ then $\{u\} \in \Delta^w$.
- For all $u, v \in \mathcal{L}(G)$ with their length larger or equal to n, u and v are in the same element of the partition iff
    - they have same subsequence of last n event occurrences and,
    - before the last event occurrence, they reach the same state

Active Sensing: $\uparrow F$ and Window Partitions

- 2-Window-Partition is $\Delta^w = \{\{\epsilon\}, \{f\}, \{d\}, \{e\}, (b,0,f), (b,0,d), (b,0,e), (d,5,c), (f,1,a), (c,1,a), (e,2,a), (a,3,e), (e,4,b), (a,4,b)\}$
- $(b,0,f)$ denotes the set of all traces $t \in \mathcal{L}(G)$ that end with subtrace bf and visit state 0 before last event f occurs



Window Partitions: Maximum Feasible Subpolicy

- Algorithm $\uparrow$F-WINDOW, given in paper
- Intuition:
  - Calculate the possible confusable pairs of elements in $\Delta^w \times \Delta^w$ under current estimation of $\Omega^{\uparrow F}$
  - Using these possible confusable pairs, remove the sensor activations that cause a violation of feasibility
  - Stop iteration when no more confusable pair can be found and no more sensor activation should be removed
- Polynomial in Δ^w

Sensor Activation Problems for the Purpose of Control

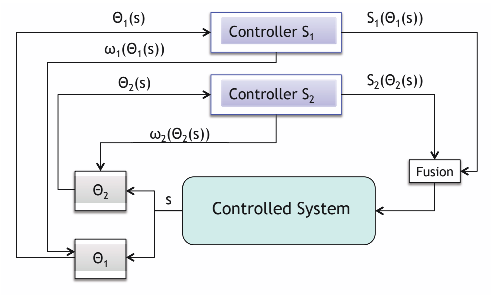


Figure: Sensor Activation for the purpose of control

Sensor Activation Problems for the Purpose of Control

- Agents activate sensors to observe event occurrences such that correct control decisions can be made
- In a decentralized system, the sufficiency of sensor activation is measured by coobservability
- With the restriction of the solution space to the transitions of the modeling automaton, polynomial algorithms are developed for calculating offline minimal sensor activation policies
- An algorithm is developed for calculating all possible minimal sensor activation policies for centralized control
- An online algorithm is developed for minimizing sensor activation, which achieves fast online calculation without any restriction of the solution space

Summary

- Different algorithms were developed for optimizing sensor activation policies that preserve diagnosability and observability for centralized systems
- Some results are extended to decentralized systems for codiagnosability and coobservability
- For a fixed number of agents, most algorithms are of worst-case polynomial complexity

Related Publications

- Journal
  - W. Wang, S. Lafortune, F. Lin, and A. R. Girard "Minimization of Dynamic Sensor Activation in Discrete Event Systems for the Purpose of Control ", to appear in IEEE Transactions on Automatic Control, whole paper
  - W. Wang, S. Lafortune, A. R. Girard, and F. Lin "Optimal Sensor Activation for Diagnosing of Discrete Event Systems", revising for resubmission to Automatica, whole paper
  - W. Wang, A. R. Girard, and C. Gong "An Algorithm for Calculating all Minimal Sensor Activation Policies in Supervisory Control", submitted to IEEE Transactions on Automatic Control

Related Publications

- Conference
  - W. Wang, S. Lafortune, A. R. Girard, and F. Lin, “Dynamic Sensor Activation for Event Diagnosis”, in Proc. 2009 American Control Conference, Jun. 2009
  - W. Wang, F. Lin, S. Lafortune, and A. R. Girard “An Online Algorithm for Minimal Sensor Activation in Discrete Event Systems”, accepted to 48th IEEE Conference on Decision and Control

Contents

- Introduction
- Dynamic Observations
- Verification of Codiagnosability under Dynamic Observations
- The Transformation of Coobservability to Codiagnosability
- Optimizing Sensor Activations
- Conclusion and Future Work

Conclusion

- If we model high level behavior of military operations by DES, then dynamic observations are indispensable
- We developed polynomial verifiers for testing codiagnosability and coobservability under transition-based dynamic observations
- We developed polynomial algorithms that transform the problem of coobservability to the problem of codiagnosability for language-based dynamic observations
- We have comprehensively investigated problems of optimizing sensor activations for both event diagnosis and control

Future Work

- Develop modular approach for modeling military missions such that optimization can be done at both global and local levels separately
- Reinvestigate dynamic observations for distributed systems for modular architecture
- Study the sensor activation problems when an activating of a sensor will sense a subset of events but cannot distinguish which event precisely

Discrete Event Modeling of Heterogeneous Human Operator Team in Classification Task

B. Hyun, C.J. Park, W. Wang and A. R. Girard

Michigan/AFRL Collaborative Center in Control Sciences
University of Michigan
Ann Arbor, Michigan

September 23, 2009

Situation Overview

- ISR mission : a team of operators performing a monitoring task that demands classification decisions.
- A stream of image data taken by UAVs shown to operators
- Operator team : mixture of two levels of expertise (novice and expert)
- Classification decision made contributes to the strategic decisions.

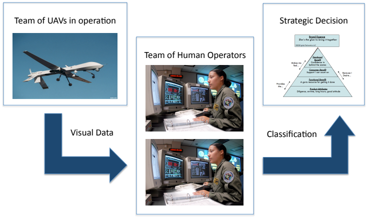


Figure: Abstracted ISR mission overview

Situation Overview

Key aspects

- Operator's mental state is related to workload. [*Yerkes-Dodson 1908, Sheridan, 1980*]
- Operator's classification decision is related to the mental state. (Role of emotional arousal in response time) [*Kuchinke et al., 2007*]
    - Supervisory action regulation strategy : when to focus and when to rest?
- Expert (trained operator) is superior to novice in some facets (decision confidence, focus, etc.)
    - Collaboration problem: what is the best way to utilize the expert?
- Human supervisory system, or a supervisor
    - Ultimately to improve the classification performance of the campaigned heterogeneous operator team
        - Coordinate individual operator actions by regulating tasks, and utilize the specialized experts
    - Sensor activation problem: when, and based on what information, does the supervisor take part?

Approach

Discrete Event System Modeling [*Cassandras & Lafortune, 1999*]

- A good candidate to represent complex human behavior in rather simple discrete fashion
- Some previous studies
  - Human-unmanned vehicle behavior for a heterogeneous unmanned vehicle system [*Nehme, 2009*]
  - Dynamic behavior (stress, fatigue, personality, cognitive complexity) filters using DESM [*Seck et al. 2005*]

- Related publication
  - Hyun, B., Park. C.J., Wang, W. Girard, A.R., "Discrete Event Modeling of Heterogeneous Human Operator Team in Classification Task", American Control Conference (ACC), 2010, Invited Session, Submitted.

Task & Rest Model

Two modes of operator activity during the campaign: Task service and Rest

- Task servicing
  - A *task* is an object of interest that is to be identified.
  - A task is modeled as a queueing process with servicing rate p (received photo).
  - Difficulty of task (Heavy or Light) by task weight $T\{H,L\}_n = n_{\{H,L\}} \cdot p$
- Rest (idle state) : queueing process
- Supervisory commands: Time-Out and Wait Flag
  - TO : authority to terminate servicing task
  - WF : allocate task if idle

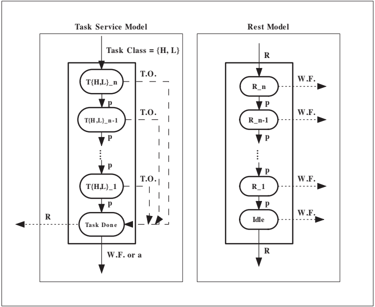


Figure: Task service & rest queueing model

Mental Model

Yerkes-Dodson Law [*Yerkes-Dodson 1908*]

- Performance as a function of mental arousal.
    - Mental arousal due to stress, boredom, hunger...
    - Mental workload vs. loading factor [*Sheridan, 1980*]

Discretized event modeling version

- Five distinct states [*Seck et al. 2005*] $X_{Me} = \{\check{s}_1, \mathring{s}_2, \mathring{s}_3, \mathring{s}_4, \check{s}_5\}$
- Event-driven transitions: a ascending, d descending
- Decision quality is determined by mental state legality

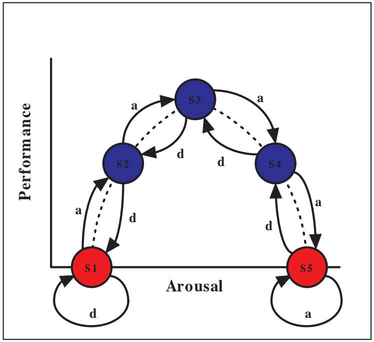


Figure: Discretized Y-D law with event-driven transitions

Monitoring Model

How the operator behaves when a task is received/not received and how the mental state is stimulated.

- Monitoring states
  - Modes of activities : Task Heavy (TH), Light (TL); Rest State (RS)
  - Intermediate auxiliary states : Task Received (TR), Task Terminated (tt)
  - $X_{Mo} = \{\underline{\underline{tt}}, TR, TL, TH, RS\}$
- Monitoring events
  - Supervisory commands : WF, TO
  - Workload and incoming photo : H, L, p
  - Mental state transitions : a, d
  - $E_{Mo} = \{WF, TO, H, L, p, a, d\}$
- Deterministic automaton $G_{Mo} = \{X_{Mo}, E_{Mo}, f, \Gamma, x_0, X_m\}$

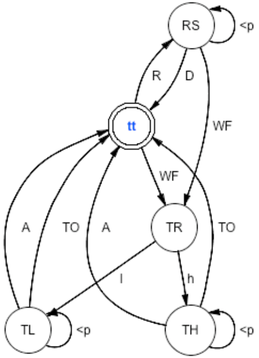


Figure: Monitoring model finite state machine

Natural Behavior : Parallel Processing

Regular, unrefined operator behavior during the campaign

- *Parallel Composition* gives a map of common-events-driven state transitions between two automata
- Mental state transition during monitoring session

$$G = G_{Me} || G_{Mo}$$

- Observed that excessive workload, or boredom, causes operator behavior to be in illegal states

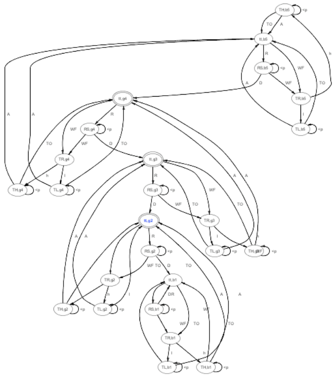


Figure: Natural behavior of a single operator (Compiled by DESUMA)

Supervisory Control: Regulated Behavior

Supervisory controller prohibits any transitions to illegal states

- Controllable events (supervisor commands) $E_c = \{TO, WF\}$
- Given E_c, natural behavior G is controllable.
- Assumed that the state machines are fully observable.

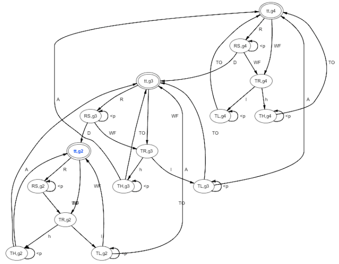


Figure: Regulated behavior for a single operator

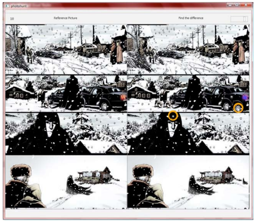
Experiment Interface Design

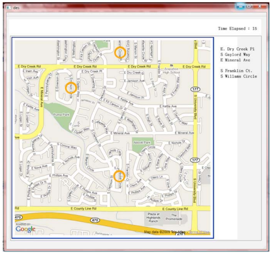
Pictorial and verbal stimuli experiment

- Human-subject experiments to investigate the mental arousal during search tasks
  - In preparation with ARC Lab members, University of Michigan
- Response time is measured to estimate the mental arousal

Future Work

- Obtain experiment results that will give some idea on mental state-workload
- Inclusion of decision confidence (subjective measure) in DESM model
- State flow simulation for dynamic scenario with uncertain data rate





Optimizing Performance of Human Operators

C. Gong, A. R. Girard, and W. Wang

Michigan/AFRL Collaborative Center in Control Sciences
University of Michigan
Ann Arbor, Michigan

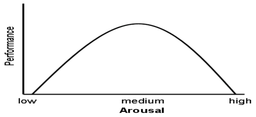
September 23, 2009

Optimizing Performance of Human Operators

Motivation & Difficulties

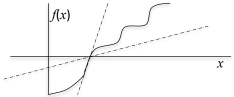
- Motivation
  - UAVs are important in warfare
  - HO are crucial in UAS
  - In future, the HO's role is decision maker
- Difficulties
  - Workload-performance curve is unknown
  - Lack of available data
  - Hard to check decision correctness

Yerkes-Dodson Law



- Arousal-performance curve
  - Concave but depends on many factors
  - Yerkes, R.M. & Dodson, J.D. (1908)

Stochastic Approximation (SA)



- Solve nonlinear eqn. $f(x) = 0$ when
  - $f(x)$ is unknown
  - Measurement $M_n = f(x_n) + U_n$, noise U_n is independent, bounded, and $E(U_n) = 0$
  - H. Robbins & S. Monro (1951)

Problem to Solve

- Given unknown concave function $f(x)$ that has attained its maximum at x*, which can be *not unique*
- Observations are the random variable $N(x)$ s.t. $E[N(x)] = f(x)$ with bounded noise
- Find a sequence $f(x_1), f(x_2), \ldots$ s.t. $f(x_n) \rightarrow f(x^*)$

Optimizing Performance of Human Operators

The Solution

- The problem can be solved by iterating

$$x_{n+1} = x_n + a_n \frac{N(x_n + c_n) - N(x_n - c_n)}{c_n}$$

- Where $\{a_n\}$ and $\{c_n\}$ are sequences of parameters that satisfy convergence requirements.

Related Publication

- C. Gong, A. R. Girard, and W. Wang, "Stochastic Approximation to Optimize the Performance of Human Operators", Submitted to American Control Conference 2010

Checking Decision Correctness

- We propose to compare the decisions of two different operators and check the consistency
- We show that optimizing the rate of such consistency is equivalent to optimizing the rate of correct decisions.

Research Goal

- Improve SA algorithms for provable highest performance of HO
  - The convergence of SA algorithms is sensitive to difference of applications
- Test algorithms in both simulations and in real UAS
- Develop adaptive control methods based on SA algorithms

A Combined Tabu Search and 2-opt Heuristic for Multiple Vehicle Routing

J. Jackson, A. Girard, S. Rasmussen, C. Schumacher

Michigan/AFRL Collaborative Center in Control Sciences
University of Michigan
Ann Arbor, Michigan

September 23, 2009

Introduction

- We introduce a combination of two methods used to solve a vehicle routing problem
- Assignment and completion order constraints are considered
- Solution procedure guarantees constraint satisfaction
- Order constraints are met even when constrained tasks are performed by different agents

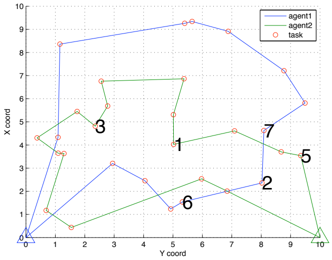


Figure: Multiple agent task assignment

Motivation

- Military missions require proper assignment of resources
- Mission planning is often rife with constraints
- These missions also require precisely enforced timing
- Solutions are needed in real time

Relevant work

Related work:

- A tabu search heuristic for vehicle routing: [M. Gentreau et al.]
- A tabu search heuristics: [F. Glover]
- Traveling salesman heuristics: [S. Lin, B. W. Kernighan], [K. Helsguan]
- UAV mission management: [M. Faied et al.], [T. Shima, S. Rasmussen]

Our work:

- Combined tabu search and 2-opt heuristic: [J. Jackson et al.]

Agents and Tasks

N_a agents are to be assigned to complete N_t tasks.

$$a_i \in \mathcal{A}, \quad i = 1, \ldots, N_a, \tag{1}$$

and

$$T_j \in \mathcal{T}, \quad j = 1, \ldots, N_t, \tag{2}$$

Pairwise precedence constraints:
$C_{p_m} = \{T_j, T_k\}$, T_j before T_k

- Transitive order constraints must be obeyed

$$p_j = \{T_k \in \mathcal{T} \mid t_k \leq t_j\}, \tag{3}$$
$$d_j = \{T_k \in \mathcal{T} \mid t_k \geq t_j\}, \tag{4}$$

Agent assignment constraints:

- Agents must be assigned from a feasible set of agents

$$\mathcal{A}_{a_j} \subseteq \mathcal{A}_{f_j} \tag{5}$$

Problem definition

Minimize the total mission time, the final arrival time of the last agent

$$\min_{\mathcal{T}a,\mathcal{C}o} \max(t_{a_{if}}) \tag{6}$$

s.t.

$$\begin{aligned}
t_k &\geq t_j \;\; \forall T_j \in p_k, \quad j = 1, \ldots, N_t, && (7)\\
t_k &\leq t_j \;\; \forall T_j \in d_k, \quad k = 1, \ldots, N_t, && (8)\\
t_k &\geq t_{a_{ik}} \;\; \forall a_i \in \mathcal{A}_{a_k}, i = 1, \ldots, N_a, && (9)\\
\mathcal{A}_{a_j} &\subseteq \mathcal{A}_{f_j}. && (10)
\end{aligned}$$

- Tasks must occur after precedents
- Tasks must occur after agents arrive
- Assigned agents must be from feasible set

Solution procedure

Assignment constraints satisfied in $\mathcal{T}a$. Assignments are in $\mathcal{P}(\mathcal{A}_{f_j})$.

$$\mathcal{T}a = \{\mathcal{A}_{a_1} \dots \mathcal{A}_{a_{N_t}}\} \quad (11)$$

- Tabu search searches assignments
- Local minima can be overcome

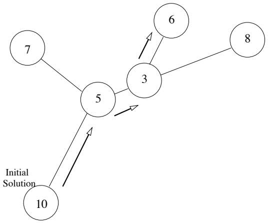
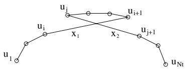


Figure: Tabu search illustration.

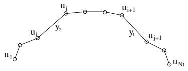
Order constraints satisfied in $\mathcal{C}o$

$$\mathcal{C}o = \{u_1 \dots u_{N_t}\}. \quad (12)$$

- 2-opt exchange operates on agent routes
- 2-opt moves proceed until improvements are exhausted



(a) Route before 2-opt move



(b) Route after 2-opt move

Figure: 2-opt illustration.

Example problem

- 2 agents
- 30 tasks

Completion order constraints

$$C_{p_1} = \{T_1, T_2\}, \quad (13)$$
$$C_{p_2} = \{T_2, T_3\}, \quad (14)$$
$$C_{p_3} = \{T_5, T_6\}, \quad (15)$$
$$C_{p_4} = \{T_6, T_7\}, \quad (16)$$

Final arrival times:
agent 1 - $33.13s$
agent 2 - $33.18s$

- Random initial solution
- Resulting cost history

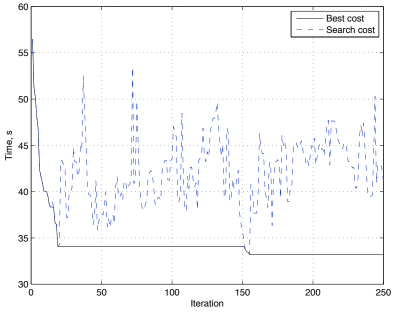


Figure: Cost history for tabu search.

Example solution

- Precedence constraints obeyed
- Algorithm enforces proper timing

Table: Completion times of constrained tasks.

Task	Time	Agent
T_1	8.26	a_2
T_2	10.51	a_1
T_3	16	a_2
T_5	3.6	a_2
T_6	7.68	a_1
T_7	12.78	a_1

Example solution

Figure: Corresponding agent routes.

Future Work

- Extend to the dynamic problem
- Dynamic addition and removal of agents and tasks
- Develop the ability to restructure constraints online

Vehicle Routing Problem Instances: Application to Multi-UAV Missions Planning

M. Faied and A. R. Girard

Michigan/AFRL Collaborative Center in Control Sciences
University of Michigan
Ann Arbor, Michigan

September 23, 2009

VRP and Multi-UAV missions

The UAV routing problem is comparable to the VRP

The VRP optimizes the routes several vehicles should follow when delivering goods to a network of customers from a single place of origin, a depot.

VRP

- Depot
- Customers

UAV missions

- Launch and Landing sites
- Targets

VRP and Multi-UAV missions

VRP and UAV missions Disparity

VRP

- One Depot
- Time Window for the VRPTW

UAV missions

- Multiple Launch and Landing sites
- Various Timing Constraints

Motivation

- Each VRP variant has its corresponding hypothetical military multi-UAV mission.
- There is a large literature on the VRP and its instances.
- Focusing on multi-objective multi-UAV mission planning problems, we aspire to take advantage of the literature in the VRP and its variants.

Motivation

- Each VRP variant has its corresponding hypothetical military multi-UAV mission.
- There is a large literature on the VRP and its instances.

- Focusing on multi-objective multi-UAV mission planning problems, we aspire to take advantage of the literature in the VRP and its variants.

VRP Instances with Application to Military Missions

Capacitated VRP

- CVRP is like VRP but every vehicle must have uniform capacity of a single commodity.

Military Application

- A bomber dropping certain number of bombs to attack ground or sea target.

VRP Instances with Application to Military Missions

Multiple Depot VRP

- A MDVRP requires the assignment of customers to depots.
- A fleet of vehicles is based at each depot.
- Each vehicle originates from one depot, services the customers assigned to that depot, and returns to the same depot.

Military Application

- This variant of VRP is represented in multiple launch and landing sites for the UAVs in the mission.

VRP Instances with Application to Military Missions

Stochastic VRP

- SVRPs are VRPs where one or several components of the problem are random.
  - Stochastic customers.
  - Stochastic demands.
  - Stochastic times.

Military Application

- Uncertainty in target locations, arrival times, or types represents this type of VRP.

VRP Instances with Application to Military Missions

Periodic VRP

- In VRPs, the planning period is a single day. In the case of the Periodic Vehicle Routing Problem (PVRP), the classical VRP is generalized by extending the planning period to M days.

Military Application

- Wide area surveillance or border patrol.

VRP Instances with Application to Military Missions

Split Delivery VRP

- SDVRP is a relaxation of the VRP wherein it is allowed that the same customer can be served by different vehicles if it reduces overall costs.

Military Application

- This instance can be viewed as a group of UAV attacking tactical targets or troop concentrations cooperatively.

VRP Instances with Application to Military Missions

VRP with Time Window

- VRPTW is VRP with a time window associated with each customer, defining an interval wherein the customer has to be supplied.

Military Application

- Rendering a detonator inoperative before it triggers an explosive device is a conventional example for VRPTW.

VRP Instances with Application to Military Missions

VRP with Pick-up and Delivery

- VRPPD is a VRP in which the customers return some commodities.

Military Application

- A good illustration for this instance is a friendly unit, which is pinned down by enemy units and needs to be rescued.

Main Contribution

- We propose an algorithm that capable of solving different variants of the VRP and also complicated UAV missions.
- We show that this algorithm can handle:
  - Basic VRP
  - MDVRP
  - VRPTW
  - A combination of MDVRP and VRPTW

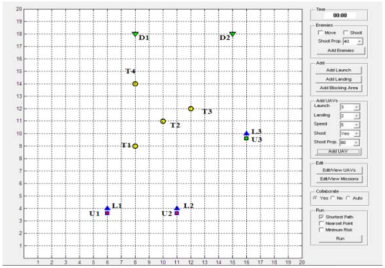


Figure: Mission Example

Main Contribution

The algorithm minimizes the VRPs cost function, and the conditions that relate to its instance and additional technical specifications that relate to the UAV mission and its targets conditions.

Conclusion

- We aim to show the correlation between the VRP with its different instances and military UAV missions.
- We introduce a brief survey on the VRP instances and associate each instance with its corresponding hypothetical military mission.
- Focusing mostly on solving complex multi-UAV mission planning problem with complex constraints, we propose an algorithm that solves the problem to optimality.

We seek to open the road for further papers that discuss this relationship and apply the ready made solutions of the VRP to multi-UAV mission planning.

Flapping Wing Micro Air Vehicles

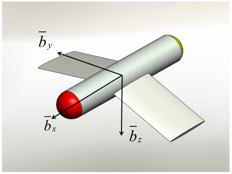
C. T. Orlowski

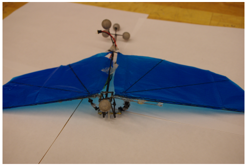
Michigan/AFRL Collaborative Center in Control Sciences
University of Michigan
Ann Arbor, Michigan

September 23, 2009

Flapping Wing Micro Air Vehicles

- Derivation and simulation of the dynamics of FWMAVs
- Control applications
  - Tail
  - Control Mass
  - Stroke Plane change
- Verification/comparison of dynamic model with prototype in VICON system





Conclusions

- Difficult problems tackled: distributed decision making with incomplete information, models of human operators, assignment problems
- Many publications: 2 journal papers accepted (one in TAC), 10 conference papers published. 3 more journal papers submitted (2 in TAC, one in Automatica).
- Two students at AFRL last summer
- Work related to MAX topics: flapping wing dynamics and control (Orlowski), scaled helicopter precision landing and health monitoring (Richardson and White), Graphical User Interfaces for scaled helicopter testbed (White), Bond Energy Algorithms for optimal path planning (Chang), Information-optimal paths (Hyun), Adversarial actions (Faied), Hardware testing (undergraduate students)