WP7 - 4:30

K. M. Passino and P. J. Antsaklis,"Artificial Intelligence Planning Problems in a Petri Net Framework" (Invited), Proc. of the 1988 American Control Conference , pp.626-631, Atlanta, Georgia, June 15-17, 1988.

# ARTIFICIAL INTELLIGENCE PLANNING PROBLEMS IN A PETRI NET FRAMEWORK

K.M. Passino and P.J. Antsaklis
Department of Electrical and Computer Engineering
University of Notre Dame, Notre Dame IN 46556

## ABSTRACT

Artificial Intelligence planning systems determine a sequence of actions to be taken to solve a problem. This is accomplished by generating and evaluating alternative courses of action. A special type of Petri net is first defined and then used to model a class of Artificial Intelligence planning problems. A planning strategy is developed using results from the theory of heuristic search. In particular, the A* algorithm is utilized. From the Petri net framework it is shown how to develop an admissible and consistent A* algorithm. As an illustration of the results three Artificial Intelligence planning problems are modelled and solved.

## 1.0 INTRODUCTION

Given a goal to achieve, an Artificial Intelligence (AI) planning system reasons from the state of its problem domain to determine the sequence of actions that will move the current state into the final goal state. Planning systems are used, for example, in the intelligent control of robots. A planner employs intelligent problem solving techniques that are fundamental to many intelligent control systems. Although many of the basic issues in planning systems are well understood empirically, they have not been adequately quantified in a mathematical framework. Formal mathematical analysis will lead to quantitative and qualitative results and it will produce modelling, analysis, and design techniques for planning systems.

In this paper an appropriate Petri net framework is introduced to quantitatively model, analyze, and design AI planning systems. Based on this Petri net framework, a planner which uses heuristic search techniques, the A* algorithm, is developed. For AI planning problems that are modelled by the Petri net it is shown that an admissible and consistent A* algorithm can be implemented. This is done by defining metric spaces associated with the Petri net and using the metric in the A* algorithm (See Section 2.3). The theoretical results are discussed in Section 2.4. To illustrate the theory, the three AI planning problems given in Section 3.0 are modelled with the Petri net, metrics are specified, and the A* algorithm is used to develop solutions. The concluding remarks given in Section 4 consist of a discussion of the examples.

In the following, some of the fundamental concepts in AI planning theory are outlined, characteristics of the Petri nets used to model the problem domain are discussed, and it is explained how heuristic search is used to implement planning strategies. Then an overview of relevant research is given.

General information on the theory of AI planning is given in [1], [2], and [14]. A very brief overview is given here to establish the terminology. An *AI planning system* consists of the planner, the problem domain, their interconnections, and the exogenous inputs. The outputs of the planner are the inputs to the problem domain. They are the control actions taken on the domain. The outputs of the problem domain are inputs to the planner. They are measured by the planner and used to determine the progress in the problem solving process. The measured exogenous input to the planner is the goal. It is the task of the planner to examine the problem domain outputs, compare them to the goal, and determine what actions to take so that the goal is met. The *problem domain* is the domain (environment) that the planner reasons about and takes actions on. One develops a model of the real problem domain to study planning systems called the *problem representation*. The functional components of a typical AI planner are as follows [1]: *Plan generation* is the process of synthesizing a set of candidate plans to achieve the current goal. This can be done for the initial plan or for *replanning* if there is a plan failure. In plan generation, the system *projects* (simulates, with a model of the problem domain) into the future, to determine if a developed plan will succeed. The system then uses heuristic *plan decision rules* based on resource utilization, probability of success, etc., to choose which plan to execute. The *plan executor* translates the chosen plan into physical actions to be taken on the problem domain. The planners considered here are ***domain independent*** since they are applicable to a variety of problem domains.

A Petri net model, defined in Section 2.1, is used for the problem representation. The definition is similar to the definition given in [12] but it also allows for control inputs to the Petri net and outputs. In [7] a "Controlled Petri Net" was defined in a somewhat different manner. The definition in Section 2 includes the so called "inhibitor arc", and equation (2.2) was created to describe its effect on the the operation of the Petri net described via state equations. The Petri net defined here also allows for the specification of a cost to fire a transition via the specification of the transition cost function in Section 2.1. Such costs could, for example, represent a measure of the resources consumed in performing the actions associated with firing a transition. Projection and plan generation are performed with the Petri net model.

The planners studied here utilize heuristic search to solve problems. The results from the theory of heuristic search using the A* algorithm outlined in Section 2.2 mainly come from [5, 6, 3, 13]. Other information can be found in [10,11]. Under certain conditions the A* algorithm can, from an initial node of a graph, find a least cost path to some goal node. When applied to planning problems the algorithm can be used for the plan decisions discussed above, to determine which plan will achieve some goal with least cost in terms of, for instance, resource consumption. Once the appropriate plan is found A* gives it to the plan executor so that the actions can be taken on the problem domain.

Other relevant research is given in [4]. There the authors use a high level Petri net to represent both the knowledge and inference strategy in expert systems. Some analysis results are obtained. Some planning systems are implemented in the computer programming language named PROLOG. An analysis of concurrency in PROLOG via Petri nets is reported in [9]. Such results could be utilized in an analysis of concurrency in AI planning systems.

## 2.0 PLANNING VIA HEURISTIC SEARCH IN A PETRI NET FRAMEWORK

In this section, the Petri net appropriate for problem domain representation is defined. Some of the theory of heuristic search using the A* algorithm is outlined. The main results on how to use the Petri net model to define a metric which can be utilized to obtain an admissible and consistent A* algorithm are given and discussed.

### 2.1 Modelling the Problem Domain with a Petri Net

A Petri Net is used to model the problem domain. Let $\mathbf{R}$ denote the set of reals and $\mathbf{R}^+$ the strictly positive reals. Let $\mathbf{N}$ denote the set of nonnegative integers. If

$x,y \in N^n$, $x=[x_1,x_2, \dots ,x_n]^t$, $y=[y_1,y_2, \dots ,y_n]^t$ (t indicates transpose) the statement $x \geq y$ is true iff $x_i \geq y_i$ $1 \leq i \leq n$. Similarly for $>,<,$ and $\leq$. For any set **X** let $|\mathbf{X}|$ denote the cardinality of **X**. A *bag* is a collection of objects over some domain **D**, but unlike standard definitions of a set, bags allow multiple occurrences of elements [12]. Let **B** be a bag, then $\#(x,\mathbf{B})$ represents the number of occurrences of element x in bag **B**. The set $\mathbf{D}^\infty$ is the set of all bags over a domain **D**. Let Ø denote the null set.

The Petri Net structure is described by $P_S=(\mathbf{P},\mathbf{T},I_D,I_N,O_D,\Delta,U,\Psi,Y)$ where:

(i) $\mathbf{P}=\{p_1,p_2, \dots ,p_n\}$ is a non-empty finite set of $n=|\mathbf{P}|$ *places* which are represented graphically with circles (○).

(ii) $\mathbf{T}=\{t_1,t_2, \dots ,t_m\}$ is a non-empty finite set of $m=|\mathbf{T}|$ *transitions* which are represented graphically by line segments or rectangles ( | , □ ).

Note that $\mathbf{P} \cap \mathbf{T} = Ø$.

(iii) $I_D:\mathbf{T} \rightarrow \mathbf{P}^\infty$ is a mapping from transitions to the set of all bags over **P**. This mapping is represented graphically by a *directed arc* ( ⟶ ) pointing from each *input place* $p_i \in I_D(t_j)$ to $t_j$.

(iv) $I_N:\mathbf{T} \rightarrow \mathbf{P}^\infty$ is a mapping from transitions to the set of all bags over **P**. This mapping is represented graphically by a *not arc* (inhibitor arc) ( —o ) pointing from each input place $p_i \in I_N(t_j)$ to $t_j$.

(v) $O_D:\mathbf{T} \rightarrow \mathbf{P}^\infty$ is a mapping from transitions to the set of all bags over **P**. This mapping is represented graphically by *directed arcs* ( ⟶ ) pointing from the transition $t_j \in \mathbf{T}$ to each *output place* $p_i \in O_D(t_j)$.

For all $p_i \in \mathbf{P}$, there exists $t_j \in \mathbf{T}$ and for all $t_j \in \mathbf{T}$ there exists $p_i \in \mathbf{P}$ such that $\#(p_i,I_D(t_j)) + \#(p_i,I_N(t_j)) + \#(p_i,O_D(t_j)) \geq 1$. That is, every arc has a transition on one end and a place on the other, and no transition or place exists without being connected to an arc.

(vi) $\Delta:\mathbf{U} \rightarrow \mathbf{T}$ is a mapping from a *control input label* $u_i \in \mathbf{U}=\{u_1,u_2, \dots , u_m\}$ $m=|\mathbf{T}|$, to a transition $t_j \in \mathbf{T}$. This mapping is represented graphically by a labelled *control arc* ( $u_i$ —□ $t_j$ ) which connects an element $u_i \in \mathbf{U}$ to a single transition $t_j \in \mathbf{T}$.

(vii) $\Psi:\mathbf{P} \rightarrow \mathbf{Y}$ is a mapping from a place $p_i \in \mathbf{P}$ to an *output label* $y_j \in \mathbf{Y}=\{y_1,y_2, \dots ,y_n\}$ $n=|\mathbf{Y}|$. This mapping is represented graphically by a labelled *output arc* ( $p_i$ ○▷— $y_j$ ) which connects an element $p_i \in \mathbf{P}$ to a single output $y_j$.

A complete description, which also includes the *execution* characteristics of the Petri Net, is given by $P_N=(P_S,X_p,X_{po},E_r,U_p,Y_p,\Phi,Z)$ where:

(i) $P_S$ is described above.

(ii) $X_p:\mathbf{P}x\mathbf{N} \rightarrow \mathbf{N}$ is the *marking function*, a mapping from a place and a nonnegative integer k representing a time step into a nonnegative integer representing the *marking* of the place. The n-dimensional column vector $x_p(k)=[X_p(p_1,k), X_p(p_2,k), \dots ,X_p(p_n,k)]^t$ is used to denote the *state* of the Petri net. The *state space* of $P_N$ is $N^n$. The state of the Petri net is represented graphically by *tokens* (●) that are put inside places (e.g. if $X_p(p_i,k)=2$ is represented as $p_i$ (●●)).

(iii) $X_{po}$ is a non-empty finite set of *initial conditions* for the state of the Petri Net; $N^n \supseteq X_{po}$.

(iv) $E_r:N^n x N^n x N \rightarrow 2^\mathbf{T}$ is the *enable rule*, a mapping from $x_p(k)$, $x_p(k+1)$, and a nonnegative integer k representing a time step into subsets of transitions that are said to be *enabled* at step k. The notation $t_j \in \mathbf{E}_r$ is used to indicate that $t_j$ is enabled at step k.

As an example, if the enable rule only depends on $x_p(k)$, then a candidate for the enable rule is given by

$$E_r(x_p(k))=\{t_j \mid X_p(p_i,k) \geq \#(p_i,I_D(t_j)), \text{ and } X_p(p_i,k)=0 \text{ if } p_i \in I_N(t_j), \text{ for all } p_i \in \mathbf{P}\} \qquad (2.1)$$

A transition can *fire* whenever it is enabled. Tokens are redistributed in the Petri net when a transition fires. The next state function below describes token movement. First, the input to the Petri net which controls the firing of the transitions is defined.

(v) $U_p:\mathbf{U}x\mathbf{N} \rightarrow \{0,1\}$ is a mapping from an input label $u_i \in \mathbf{U}$ and a nonnegative integer k representing a time step to 0, indicating that the transition $t_j \in \Delta(u_i)$ cannot fire, or to a 1 indicating that a transition $t_j \in \Delta(u_i)$ is to fire ($t_j \in \mathbf{E}_r$). The vector $u_p(k)=[U_p(u_1,k),U_p(u_2,k), \dots ,U_p(u_m,k)]^t$ is the *control input* to the Petri net. Let $\mathbf{U}_p$ be the set of column vectors of an mxm identity matrix. Only one transition is allowed to fire at once, therefore $u_p(k) \in \mathbf{U}_p$ for all k. For example, if $u_p(k)=[0\ 0 \dots 0\ U_p(u_i,k)\ 0 \dots 0\ 0]^t$, and $U_p(u_i,k)=1$ is in the jth position, $1 \leq j \leq m$, then transition $t_j \in \mathbf{E}_r$ ($t_j \in \Delta(u_i)$) is fired. If $U_p(u_i,k)=0$ then no transition is fired.

(vi) $Y_p:\mathbf{Y}x\mathbf{N} \rightarrow \mathbf{N}$ is a mapping from an output label $y_j \in \mathbf{Y}$ and nonnegative integer k representing a time step to $X_p(p_i,k)$ such that $y_j \in \Psi(p_i)$. The n-dimensional column vector $y_p(k)=[Y_p(y_1,k),Y_p(y_2,k), \dots ,Y_p(y_n,k)]^t$ is the *output* of the Petri net and $y_p(k) \in \mathbf{N}^n$ for all k.

(vii) $\Phi:N^n x \mathbf{U}_p x \mathbf{N} \rightarrow N^n$ is the *next state function*, a mapping from the current state $x_p(k)$, a control input vector $u_p(k)$ that *fires* a transition $t_j \in \mathbf{E}_r$, and a nonnegative integer k representing a time step, into the next state $x_p(k+1)$. The next state function is defined iff for a control input $u_p(k) \in \mathbf{U}_p$ the associated transition $t_j \in \mathbf{T}$ is enabled at step k.

(viii) $Z:\mathbf{T}xN^n xN^n x\mathbf{R}^+ \cup \{0\} \rightarrow \mathbf{R}^+$ is the *transition cost function*, a mapping from a transition $t_j \in \mathbf{E}_r$, $x_p(k)$ and $x_p(k+1)$, and a nonnegative real number into a strictly positive real number that represents the *cost* of firing $t_j$. Since the firing of a transition often represents some computation or action performed Z is a measure of the cost to process the state $x_p(k)$ into $x_p(k+1)$ by firing $t_j$. The transition cost function is defined iff the transition $t_j \in \mathbf{T}$ is enabled at step k. Furthermore, Z is chosen so that for all $r \in \mathbf{R}^+ \cup \{0\}$, $Z(t_j,x_p(k),x_p(k+1),r)>r$. The significance of the value of r will be discussed in the next section.

As an example of a next state function, consider one that depends only on the current state and transition fired. Let $\Phi=[\phi_1,\phi_2, \dots ,\phi_n]^t$. Let $A^-=[a^-_{ij}]$, where $a^-_{ij}=\#(p_i,I_D(t_j))$ and $A^+=[a^+_{ij}]$, where $a^+_{ij}=\#(p_i,O_D(t_j))$. Let $A=A^+ - A^-$. Let

$B=[b_{ij}]$ where $b_{ij}=0$ if $t_j \notin I_N(p_i)$ and $b_{ij}=1$ if $t_j \in I_N(p_i)$. Let $a^-_j$, and $b_j$ refer to the jth columns of $A^-$ and B respectively. Then the example enable rule (2.1) can be stated as

$$E_r(x_p(k))=\{t_j \mid x_p(k)\geq a^-_j \text{ and } x_p(k)^t b_j=0\} \qquad (2.2)$$

If $u_p(k)$ is chosen so that $t_j \in E_r$ is fired, where $E_r$ is given by (2.2), then

$$\phi_i=X_p(p_i,k+1)=X_p(p_i,k) - \#(p_i,I_D(t_j)) + \#(p_i,O_D(t_j)) \text{ for all } i,\ 1\leq i\leq n \qquad (2.3)$$

and,

$$x_p(k+1)=x_p(k) + Au_p(k) \qquad (2.4)$$
$$y_p(k)=x_p(k)$$

which are the *state equations* describing the Petri net [12].

Next the some of the results from the theory of heuristic search using the $A^*$ algorithm are outlined. These results will be combined with the Petri net model to obtain the results in Section 2.3.

## 2.2 Heuristic Search: The $A^*$ Algorithm

Some results from the theory of a heuristic search, in particular the $A^*$ algorithm, are outlined below [5, 6, 3, 13].

The problem space is represented explicitly by a δ-Graph where:

(i) $X=\{x_1,x_2,x_3, \ldots \}$ is the non-empty possibly infinite set of nodes.
(ii) $E=\{e_{ij}\}=\{(x_i,x_j) \mid x_i,x_j \in X\}$ is the non-empty possibly infinite set of directed arcs pointing from $x_i$ to $x_j$.
(iii) $C=\{c_{ij}\}=\{c_{ij} \mid e_{ij}\in E\}$ is the non-empty possibly infinite set of costs associated with each arc. Also, for all $c_{ij}\in C$, $c_{ij}\geq\delta>0$.

An *implicit* representation of the δ-Graph **G** is given by a set of source nodes and a *successor operator* $\Gamma: X\rightarrow 2^{X\times C}$. When Γ is applied to a node x it is *expanded*. The δ-Graph is generated by repeated application of the successor operator Γ to nodes that are generated in expanding nodes.

The *subgraph* $G_x$ from any $x\in X$ is the graph defined implicitly by a single source node x and some Γ defined on X. Each node in $G_x$ is *accessible* from x. A *path* from $x_1$ to $x_k$ is an ordered set of nodes $<x_1,x_2, \ldots ,x_k>$ such that $x_{i+1}\in\Gamma(x_i)$ for all $1\leq i\leq k-1$. There exists a path from $x_i$ to $x_j$ iff $x_j$ is accessible from $x_i$. Every path has a cost which is obtained by adding the costs of each arc $c_{i,i+1}\in C$. An *optimal path* from $x_i$ to $x_j$ is a path having the smallest cost over the set of all paths from $x_i$ to $x_j$, call this cost $h(x_i,x_j)$. Denote an estimate of this cost by $\hat{h}(x_i,x_j)$. The concern is with the subgraph $G_{x_0}$ from a single specified *start node* $x_0\in X$. Define the non-empty set $X_g$, $X\supseteq X_g$ of nodes in $G_{x_0}$ as *goal nodes*. For any node x in $G_{x_0}$ an element $x_g\in X_g$ is a *preferred* goal node of x iff the cost of the optimal path from x to $x_g$ does not exceed the cost of any other path from x to any other member $x'\in X_g$.

The objective is to find the optimal path from the start node to preferred goal node. To help guide the search an *evaluation function* $v: X\rightarrow R^+\cup\{0\}$ is used to rank how promising it is that a node is on an optimal path. The evaluation function is defined so that the node with the smallest value of v(x) is chosen for expansion. One algorithm that performs heuristic search is the $A^*$ algorithm which is now defined:

Let $L_c$ be the set of information about ("closed") nodes which have been expanded, and $L_o$ the set of information about ("open") nodes which are candidates for expansion. The elements of $L_c$ and $L_o$ are triples $(x',v(x'),x)$ where $x'\in\Gamma(x)$ and x represents a *pointer* from x' to x. The notation $x'\in L_c$, $x'\in L_o$ is used to reference element $(x',v(x'),x)$.

<u>The $A^*$ Algorithm</u>:
Set $L_c=\emptyset$ and $L_o=\{x_0\}$, the start node.
*While* $|L_o|>0$ *do*
  Choose $x\in L_o$ so that x has the lowest value of v(x) (Resolve ties arbitrarily).
  *If* $x\in X_g$,
    *then* Exit with success and the solution path is found by tracing back through the pointers,
  *else* Place x in $L_c$.
  *For* each $x'\in\Gamma(x)$ *do*
    Calculate v(x').
    (i) *If* $x'\notin L_c$ and $x'\notin L_o$ *then* place $(x',v(x'),x)$ in $L_o$.
    (ii) *If* $x'\in L_o$ *then* denote this x' with $(x',v_1(x'),x_1)$, where $v_1(x')$ was the value of the evaluation function, and $x_1$ was the parent node of x', computed in a previous step. If $v(x')<v_1(x')$ then replace $(x',v_1(x'),x_1)$ with $(x',v(x'),x)$ in $L_o$.
    (iii) *If* $x'\in L_c$ *then* denote this x' with $(x',v_1(x'),x_1)$, where $v_1(x')$ was the value of the evaluation function, and $x_1$ was the parent node of x', computed in a previous step. If $v(x')<v_1(x')$ then remove $(x',v_1(x'),x_1)$ from $L_c$ and place $(x',v(x'),x)$ in $L_o$.
  *end*
*Exit* with failure.

The evaluation function v(x) must be chosen. Let f(x) be the actual cost of an optimal path constrained to go through x from the start node $x_0$ to a preferred goal node $x_g\in X_g$. Let $f(x)=g(x)+h(x)$ where g(x) is the actual cost of an optimal path from $x_0$ to x and h(x) is the cost of an optimal path from x to a preferred goal node of x, i.e., $h(x)=\min_{x_g\in X_g}(h(x,x_g))$. Since f(x), g(x), and h(x) are not known apriori, the estimates $\hat{f}(x)$, $\hat{g}(x)$, and $\hat{h}(x)$ are used. Therefore, choose $v(x)=\hat{f}(x)=\hat{g}(x)+\hat{h}(x)$. The function $\hat{h}(x)$, called the *heuristic* component of the evaluation function, is used to capture information from the problem domain to guide the search. If $\hat{h}(x)$ satisfies certain properties then the $A^*$ algorithm performs well.

If some goal node is accessible from the start node and $0\leq\hat{h}(x)\leq h(x)$ for all $x\in X$, then $A^*$ is *admissible* i.e., it is guaranteed to find an optimal path from the start node to a preferred goal node for any δ-Graph. The heuristic $\hat{h}(x)$ is said to be *consistent* if $h(x_i,x_j)+\hat{h}(x_j)\geq\hat{h}(x_i)$ for all $x_i,x_j\in X$. The heuristic $\hat{h}(x)$ is said to satisfy a *monotone restriction* if for all $x_j\in\Gamma(x_i)$, $x_i\in X$, $h(x_i,x_j)+\hat{h}(x_j)\geq\hat{h}(x_i)$. If $\hat{h}(x)$ is consistent then it automatically satisfies the monotone restriction. If $\hat{h}(x)$ satisfies the monotone restriction then (i) if $A^*$ selects x for expansion $\hat{g}(x)=g(x)$, and the value of $\hat{f}(x')$ for x' on the path from $x_0$ to x is nondecreasing; consequently (ii) step (iii) in the $A^*$ algorithm is vacuous and can be removed. Suppose that there are two versions of the $A^*$ algorithm called $A_1$ and $A_2$ which use evaluation functions $\hat{f}_i(x)=\hat{g}_i(x)+\hat{h}_i(x)$ where $0\le\hat{h}_i(x)\le h(x)$ for all $x\in X$, $i=1,2$. The algorithm $A_2$ is said to be *more informed* than $A_1$ if for all nongoal nodes x, $\hat{h}_2(x)>\hat{h}_1(x)$. The following *optimality* result was obtained. If $\hat{h}_2(x)>\hat{h}_1(x)$ then at the termination of their searches every node expanded by $A_2$ was also expanded by $A_1$. It follows that $A_1$ expands at least as many nodes as does $A_2$.

### 2.3 Heuristic Search in a Petri Net Framework

Utilizing these results on heuristic search outlined in Section 2.2 and the Petri net model defined in Section 2.1 it is shown how to develop an admissible and consistent $A^*$ algorithm for a certain class of problems. First, a metric and a metric space is defined [8].

Let $\bar{X}$ be an arbitrary non-empty set and let $\rho$: $\bar{X}x\bar{X}\rightarrow\mathbf{R}$ where $\rho$ has the following properties:

(i) $\rho(x,y)\ge0$ for all $x,y=\bar{X}$ and $\rho(x,y)=0$ iff $x=y$,

(ii) $\rho(x,y)=\rho(y,x)$ for all $x,y=\bar{X}$,

(iii) $\rho(x,y)\le\rho(x,y)+\rho(z,y)$ for all $x,y,z=\bar{X}$ (Triangle Inequality).

The function $\rho$ is called a *metric* on $\bar{X}$ and the mathematical system consisting of $\rho$ and $\bar{X}$, denoted $\{\bar{X};\rho\}$, is called a *metric space*. Equivalently, $\rho$ is a metric iff, (i) $\rho(x,x)=0$ iff $x=y$, and (ii) $\rho(y,z)\le\rho(x,y)+\rho(x,z)$ for all $x,y,z\in\bar{X}$.

The next theorem says that if the nodes of a certain δ-Graph and the heuristic function $\hat{h}(x_i,x_j)$ form a metric space, then the $A^*$ algorithm is both admissible and consistent.

Theorem 1: Define $\eta_{ij}$:$\mathbf{R}^+\cup\{0\}\rightarrow\mathbf{R}^+$ and $\eta_{ij}(x)>x$ for all $x\in\mathbf{R}^+\cup\{0\}$. Let $G=(X,E,C)$ be a δ-Graph where for all $c_{ij}\in C$, $c_{ij}=\eta_{ij}(\hat{h}(x,x'))>\delta>0$ with $x'\in\Gamma(x)$. If $\{\bar{X};\hat{h}(x_i,x_j)\}$ is a metric space then $A^*$ is both admissible and consistent.

Proof: For admissibility it must be shown that $0\le\hat{h}(x_i,x_g)\le h(x_i,x_g)$ for all $x_i\in X$, $x_g\in X_g$. Since $\hat{h}(x_i,x_g)$ is a metric $\hat{h}(x_i,x_g)\ge0$ so all that must be shown is $\hat{h}(x_i,x_g)\le h(x_i,x_g)$. Let $<x_0,x_1, \dots ,x_k>$ be a path generated by $A^*$. From the triangle inequality, $\hat{h}(x_i,x_k)\le\hat{h}(x_i,x_{i+1})+\hat{h}(x_{i+1},x_k)$ for all i, $0\le i\le k-1$.Therefore

$$\hat{h}(x_i,x_k)\le\sum_{i=0}^{k-1}\hat{h}(x_i,x_{i+1}).$$

Selecting $\eta_{ij}$ as stated in theorem above, it follows that

$$\hat{h}(x_i,x_k)\le\sum_{i=0}^{k-1}\eta_{i,i+1}(\hat{h}(x_i,x_{i+1}))=h(x_i,x_k).$$

By assumption, the node $x_g$ is accessible, therefore for some path generated by $A^*$, $x_k=x_g$, hence $A^*$ is admissible. To prove consistency it must be shown that $h(x_i,x_j)+\hat{h}(x_j)\ge\hat{h}(x_i)$ for all $x_i,x_j\in X$. By the triangle inequality $\hat{h}(x_i,x_j)+\hat{h}(x_j,x_g)\ge\hat{h}(x_i,x_g)$ and from admissibility $h(x_i,x_j)\ge\hat{h}(x_i,x_j)$ so that $h(x_i,x_j)+\hat{h}(x_j,x_g)\ge\hat{h}(x_i,x_g)$. Note that the monotone restriction is also satisfied. QED

Note that for $\mathbf{R}^n\supseteq\bar{X}$ if $\{\mathbf{R}^n;\rho\}$ is a metric space then so is $\{\bar{X};\rho\}$ [8]. A few candidate metrics are listed below:

(i) $\rho_d(x,y)=0$ if $x=y$ and 1 if $x\neq y$ is a metric on $\bar{X}$ (an arbitrary non-empty set) called the *discrete metric*.

Denote elements $x,y\in\mathbf{R}^n$ by $x=[\xi_1,\xi_2, \dots ,\xi_n]^t$ and $y=[\lambda_1,\lambda_2, \dots ,\lambda_n]^t$ where $\xi_i,\lambda_i\in\mathbf{R}$ for $i=1,2, \dots ,n$.

(ii) Let $\bar{X}=\mathbf{R}^n$ and $p\in\mathbf{R}$, $1\le p\le\infty$, then $\{\mathbf{R}^n;\rho_p\}$ is a metric space where

$$\rho_p(x,y)=\left[\sum_{i=1}^{n}|\xi_i-\lambda_i|^p\right]^{1/p} \tag{2.5}$$

Let W be a positive definite matrix. Then if

$$\rho_2(x,y)=\left[(x-y)^tW(x-y)\right]^{1/2} \tag{2.6}$$

$\{\mathbf{R}^n;\rho_2\}$is a metric space.

(iii) Let $\bar{X}=\mathbf{R}^n$ and $x,y\in\mathbf{R}^n$, and

$$\rho_\infty(x,y)=\max\{|\xi_1-\lambda_1|,|\xi_2-\lambda_2|, \dots ,|\xi_n-\lambda_n|\} \tag{2.7}$$

then $\{\mathbf{R}^n;\rho_\infty\}$ is a metric space.

The next theorem is an application of Theorem 1.

Theorem 2: Suppose that a system is described with the Petri net $\mathbf{P_N}$ and that an initial state and a reachable desired state are specified. Then there exists an admissible and consistent $A^*$ algorithm that can select the appropriate sequence of transition firings to move the initial state to the desired state with least cost.

Proof: First, a δ-Graph representation of the Petri net $\mathbf{P_N}$ is given. Let $\mathbf{X}=\mathbf{N}^n$, the state space of the Petri net $\mathbf{P_N}$. Suppose that the enable rule $\mathbf{E_r}$ is given by (2.2) and the next state function by (2.4). Let $|\mathbf{X_g}|=1$ and $x_g$ denote the desired state (goal node). The start node $x_0=x_p(0)\in\mathbf{X_{p0}}$ and the edges and costs are generated by

$$\Gamma(x_p(k))=\{(x_p(k+1),c)|\ x_p(k+1)=x_p(k)+Au_p(k),\ t_j\in\mathbf{E_r} \text{ and } c=Z(t_j,x_p(k),x_p(k+1),r)\} \tag{2.8}$$

with $r=\hat{h}(x_p(k),x_p(k+1))$. Note that $|\Gamma(x_p(k))|$ is finite for all $x_p(k)$ and the assigned cost $c>\delta>0$. The assigned cost depends on the value of the metric $\hat{h}(x_p(k),x_p(k+1))$. Note that for a particular Petri net model if it is the case that $x_p(k)\neq x_p(k+1)$ for all k then Theorem 1 is also valid for $\eta_{ij}(r)\ge r$ for all $r\in\mathbf{R}$. In this case the transition cost function can be chosen as $Z(t_j,x_p(k),x_p(k+1),r)\ge r$ and used in (2.8).

Next, choose $\bar{X}=X$, and $\hat{h}(x_i,x_j)$ equal to any valid metric such as $\rho_d$, $\rho_p$, $\rho_2$, or $\rho_\infty$. Then by Theorem 1, $A^*$ is both admissible and consistent and thus finds a least cost path. To choose the sequence of transition firings $A^*$ traces back through the pointers from $x_g$ to $x_0$. QED

### 2.4 Discussion

The significance of the results contained in the two theorems is discussed here.

Does Theorem 1 provide a method to pick the value of the heuristic function $\hat{h}(x_i,x_j)$ to obtain admissibility and consistency for any δ-Graph? No. There is a restriction placed on the costs allowed that depends on the choice of the metric. However, costs may be able to be redefined to fit the particular problem at hand and Theorem 1. The flexibility in the choice of the metric will help here. Another question left to be answered is how good the metrics are? This is a standard problem in the theory of the $A^*$ algorithm. For instance, if the chosen metric is such that it gives an extremely conservative estimate of the cost from all nodes x to a the preferred goal node, then A* may expand too many nodes in finding a solution. The computational complexity involved in computing the metric itself must also be considered when studying the computational demands of a particular A* algorithm.

Theorem 2 allows the planning system designer to transfer the work of choosing the heuristic function $\hat{h}(x_i,x_j)$ to forming the Petri net model of the problem domain under consideration. This can be valuable if it is not clear how to pick the heuristic function for a particular problem domain. However, if the problem domain cannot be modelled via the Petri net defined the result cannot be utilized. Also, practically speaking, the Petri net model may be too complex to be utilized in the implementation of the A* algorithm.

## 3.0 EXAMPLES

This section contains three examples to illustrate some of the results in Section 2. These include the blocks world planning problem, a "think and jump" game, and the missionaries and cannibals problem. Many details were omitted to save space.

Blocks World: The first example is the so called "blocks world" [1]. In this classic AI planning problem there are three blocks labelled "a", "b", and "c" which are placed on a table "t". Using a robot arm, the objective is to move the blocks, by stacking and unstacking them, so that from an initial configuration of the blocks, a final desired configuration can be obtained. Let **ab, ct,** and **cab** represent the facts that "block **a** is on **b**", "block **c** is on the table", and "block **c** is on **a** which is on **b**" respectively. Also let **abt;ct** represent the fact that block **a** is on block **b** which is on the table and block **c** is on the table. The initial configuration is **ca;bt**, and the desired one is **abct**. A Petri net model was constructed. This net had 9 places, 18 transitions and used inhibitor arcs. A planning strategy was specified by choosing the metric $\hat{h}(x_i,x_j)=\rho_2(x_i,x_j)$ defined in (2.6) with $W=I_{9x9}$, the 9x9 identity matrix. The A* algorithm was implemented to generate the sequence of actions to be taken by the planner. A* expanded 5 nodes before it found the optimal path. The sequence of positions of blocks considered in solving the problem (expanded by A*) was: **cat;bt, cat;bc, at;bt;ct, cbt;at, bct;at**, then the solution was found. The solution was optimal relative to the number of steps required.

Notice that W is a parameter to be chosen to load heuristic information into the A* algorithm. Let $W=[w_{ij}]$ where $w_{ij}=0$ if $i\neq j$ and $w_{ii}>0$ for all i. Each *weight* $w_{ii}$ corresponds to a place $p_i$. If it is important for tokens to be in a place $p_i$ then the corresponding value of $w_{ii}$ should chosen to be small relative to $w_{kk}$ where $k\neq i$. If it is important not to have tokens in a particular place $p_i$ then $w_{ii}$ should be chosen large relative to $w_{kk}$ where $k\neq i$. A particular set of weights was chosen for blocks world to reflect the importance of achieving the goal state, and that it may be important to unstack the blocks to solve the problem. When A* is executed with this new choice for W it expanded only 4 nodes in finding a solution: **cat;bt, cat;bc, at;bt;ct, bct;at**.

Think and Jump Game: The second example is a "think-and-jump" game involving a triangular board with ten holes in it, and 9 pegs which fit into the holes. The 9 pegs are put in the holes. Pegs are removed if they are "jumped" by other other pegs. A peg can jump another peg only if there is an empty hole directly on the other side of the peg. See Figure 3.1.

First, the Petri net model was constructed by letting the places $p_i$ correspond to the holes i in the board and the

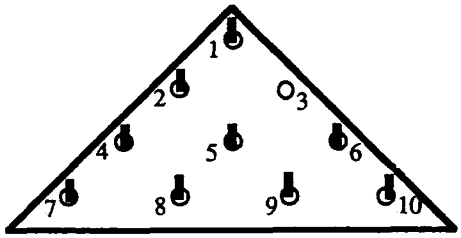


Figure 3.1 Think and Jump Game
(A version of Chinese checkers)

tokens correspond to the pegs. The initial configuration of pegs is to have one in all the holes except for hole 3, and the goal configuration is to have all pegs removed from the board except for one in hole 2. The heuristic function is chosen to be $\hat{h}(x_i,x_j)=\rho_2(x_i,x_j)$ where W is a diagonal matrix with weights chosen to reflect the importance of attaining a token in places $p_4,p_5,p_7$, and $p_9$ since in the step before solving the problem, the algorithm must have tokens in two of these places. The function Z can also be used to capture heuristic information about the problem domain. A heuristic used in this problems solution is to try to keep the pegs in the middle of the board. To capture this heuristic information high values of the cost function are assigned to transitions that fire and move tokens from the middle of the board. For these choices A* was used to find a solution to the think and jump problem above. It expanded 58 nodes. The solution generated by the planner used only 8 jumps.

Missionaries and Cannibals Problem: Three missionaries and three cannibals are trying to cross a river. As their only means of navigation, they have a small boat, which can hold one or two people. If the cannibals outnumber the missionaries on either side of the river, the missionaries will be eaten; this is to be avoided. Find a way to get them all across the river.

For the Petri net construction $\mathbf{P}=\{p_i\}$, i=1,2, ... ,6, and $\mathbf{T}=\{t_j\}$, j=1,2, ... ,10 and the net is given in Figure 3.2.

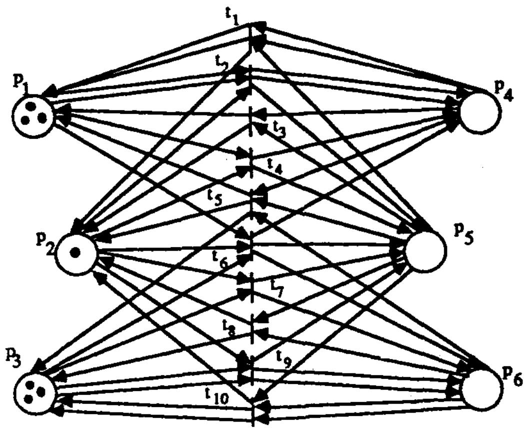


Figure 3.2 Petri Net Model of the Missionaries and Cannibals Problem

Let tokens in: (i) $p_1$ ($p_4$) represent that cannibals are on the left (right) side of the river, (ii) $p_2$ ($p_5$) represent that the boat is on the left (right) side of the river, (iii) $p_3$ ($p_6$) represent that missionaries are on the left (right) side of the river. The initial state is $x_p(0)=[3\ 1\ 3\ 0\ 0\ 0]^t$, and the goal state is $x_g=[0\ 0\ 0\ 3\ 1\ 3]^t$. The next state function (2.3) and the state equations (2.4) were used.

Note that the above Petri net graph does not represent the complete problem. The fact that cannibals cannot outnumber missionaries is not yet represented. Rather than using the graphical representation for this fact, this information is loaded directly into the enable rule. Therefore, choose

$$\mathbf{E}_r(x_p(k),x_p(k+1))=\{t_j \mid x_p(k)\geq a^-_j, \text{ and } X_p(p_3,k+1)=X_p(p_6,k+1)=0 \text{ or}$$
$$\text{if } X_p(p_3,k+1)>0 \text{ then } X_p(p_3,k+1)\geq X_p(p_1,k+1),$$
$$\text{if } X_p(p_6,k+1)>0 \text{ then } X_p(p_6,k+1)\geq X_p(p_4,k+1)\}$$

The transition cost function $Z(x_p(k),x_p(k+1),r)=r$ for all $r\in \mathbf{R}$, and this is valid since $x_p(k)\neq x_p(k+1)$ for all k.

The heuristic function is chosen to be $\hat{h}(x_i,x_j)=\rho_2(x_i,x_j)$ where $W=I_{6x6}$, the 6x6 identitiy matrix. $A^*$ expanded 13 nodes in determining the solution. The solution generated by the planner is the sequence of transition firings: $t_6,t_8,t_2,t_3,t_9,t_5,t_9,t_3,t_2,t_8,t_6$. Thesymmetry in the solution sequence is interesting. The solution involves 11 boat trips which is the minimum number of trips needed to solve the problem.

## 4.0 CONCLUDING REMARKS

The examples given in the Section 3.0 are discussed here briefly. In the blocks world planning problem the Petri net graph that was generated was omitted due to space limitations. Thhe metric was chosen via the guidelines given by the Theorems; but there is still much flexibility allowed in capturing heuristic information about the problem domain since any valid metric is allowed. In this example it was demonstrated that an appropriate choice for the metric (via W) can lead to a more efficient A* algorithm. In the think and jump problem it was shown how to use the transition cost function to capture heuristic information about the problem domain. Notice that this provides another way to capture heuristic information about the problem domain. Hence, the metric can be used to capture heuristic information pertaining to a desire to have tokens in places and the transition cost function can be chosen to capture heuristic information pertaining to the desire to fire a transition. The missionaries and cannibals example demonstrated the flexibility of the Petri net model. All characteristics of the problem need not be modelled with the Petri net graph since certain information can be captured in, for instance, the enable rule of the Petri net.

## 5.0 REFERENCES


[1] Charniak E., McDermott D.,Introduction to Artificial Intelligence Addison Wesley, Reading Mass, 1985.

[2] Cohen P.R., Feigenbaum E.A.,The Handbook of Artificial Intelligence, Volume 3, Kaufmann,CA, 1982.

[3] Gelperin D., "On the Optimality of A*", Artificial Intelligence, Vol. 8, pp 69-76, 1977.

[4] Giordana A., Saitta L., "Modelling Production Rules by Means of Predicate Transition Networks", Inf.Sciences, Vol. 35, No. 1, 1985.

[5] Hart P.E., Nilsson N.J., Raphael B.,"A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-4, No. 2, July 1968.

[6] Hart P.E., Nilsson N.J., Raphael B.,"Correction to: A Formal Basis for the Heuristic Determination of Minimum Cost Paths", SIGART Newsletter Vol. 37, pp 28-29, 1972.

[7] Krogh B. "Controlled Petri Nets and Maximally Permissive Feedback Logic", Proc. 1987 Allerton Conf. Urbana, IL, Sept. 1987.

[8] Michel A.N., Herget C.J., Mathematical Foundations in Engineering and Science: Algebra and Analysis, Prentice Hall, NJ, 1981.

[9] Murata T., Zhang D., "A High-Level Petri Net Model for Parallel Interpretation of Logic Programs", IEEE Conf. on Computer Languages, Oct. 1986.

[10] Nilsson N.J., Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, NY, 1971.

[11] Nilsson N.J., Principles of Artificial Intelligence, Tioga, NY, 1980.

[12] Peterson J., Petri Net Theory and the Modeling of Systems, Prentice Hall, NJ, 1981.

[13] Vanderbrug G.J., "Problem Representations and Formal Properties of Heuristic Search", Information Sci., Vol. 11, pp 279-307, 1976.

[14] Wilensky R., Planning and Understanding, Addison Wesley, Reading, Mass., 1983.