NETWORK LANGUAGES FOR CONCURRENT MULTIAGENT SYSTEMS

D R A F T*

Boris Stilman
Department of Computer Science & Engineering
University of Colorado at Denver, Campus Box 109, Denver, CO 80217-3364, USA
Email: bstilman@cse.cudenver.edu

Abstract: This paper** reports new results in the development of Linguistic Geometry towards concurrent multiagent aerospace systems. This formal theory is intended to discover the inner properties of human expert heuristics, which have been successful in a certain class of complex control systems, and apply them to different systems. The Linguistic Geometry relies on the formalization of search heuristics of the highly-skilled human experts, which allow for the decomposition of a complex system into a dynamic hierarchy of subsystems, and thus solve intractable problems by reducing the search dramatically. In this paper we consider briefly the Linguistic Geometry tools and their application to multiagent systems represented 2D optimization problems for autonomous robotic vehicles in aerospace environment. First, we consider only serial motions, then we relax these constraints by allowing the cooperating agents move simultaneously if necessary while the motions of opposing agents alternate. Next we relax these constraints completely by allowing all the agents move simultaneously if necessary. A comparison of the searches for serial and concurrent cases shows that search reduction achieved in all types of problems with concurrent motions is even more dramatic than it was in the serial one.

Keywords: Artificial Intelligence, Multiagent Systems, Linguistic Geometry, Heuristic Search, Network Languages, Serial and Concurrent Motions.

1. Introduction

Aerospace problems such as long and short-range mission planning, especially for autonomous navigation, scheduling, aerospace robot control, long-range satellite service, aerospace combat operations control, etc. can be formally represented as reasoning about complex large-scale control systems.

A military objective in strategic warfare is to effectively detect missile launches. Although satellites can detect the physical features of a launch, the false-positive error rate is higher than desirable. An effective strategy is to use a reconnaissance aircraft to further investigate launches reported by a satellite's sensors. An advantage of this approach is that an aircraft has the flexibility to respond appropriately, rather than just detect. For example, if a launch site is found, the aircraft can destroy it; if no launch site is found where it was expected, then the aircraft can search in a distance-bounded area for a mobile launcher in transit.

For reconnaissance, aircraft piloted by autonomous agents have certain advantages including less risk to and less utilization of human pilots and non-susceptibility to jamming in the case of remotely piloted aircraft. However, if we are to use an autonomous reconnaissance aircraft, the algorithms which implement it must include those for decision-making while planning a route

---

* The final version will be published as Stilman, B., Network Languages for Intelligent Control, *An International Journal: Computers & Mathematics with Applications*, 1996.

** Supported in part by the **U.S. Air Force Office of Scientific Research** through **Phillips Lab, Kirtland AFB**, NM, USA, and by **Sandia National Laboratories**, Albuquerque, NM, USA.

through territory guarded by aggressive counterparts. The problems described in this paper pit two adversarial aircraft against two autonomous aircraft, the so-called unmanned aerial vehicles (UAV), on a mission to reconnoiter supposed launch sites. Similar problems of generating and real time replanning of the combat scenarios are essential for the Navy and Army combat.

Problems of this type are usually described mathematically in the form of pursuit-evasion differential games. The classic approach based on the theory of Differential Games is insufficient, especially in case of dynamic, multiagent models [3]. It is well known that there exists a small number of differential games, for which exact solutions are available. There are a few more for which numerical solutions can be computed, under rather restrictive conditions, in a reasonable amount of time. It is even worse that each of these games is one-to-one which is very far from the real world combat scenarios. They are also of the "zero-sum type" which does not allow a new agent to join the game or all the agents of both sides to be disengaged. Another difficulties arise from the requirements of the 3D modeling and from limitation of the lifetime of the agents.

Following [1, 4] discrete-event modeling of complex control systems can be implemented as a purely interrogative simulation. These techniques can be based on generating geometrically meaningful states rather than time increments with due respect to the timeliness of actions. By discretizing time, a finite game tree can be obtained. The nodes of the tree represent the states of the game, where the players can select their controls for a given period of time. It is also possible to distinguish the respective moves of the two sides (including simultaneous actions). Thus, the branches of the tree are the moves in the game space. The pruning of such tree is the basic task of heuristic search techniques. Interrogative approach to control problems offers much faster execution and clearer simulator definition [2]. Several alternative search methods are applicable to this type of problem. However, in all cases, especially for concurrent systems, the branching factor for search is unacceptably large unless very good domain knowledge is used to direct search. All known, successful AI programs that address large problems make heavy use of domain information to guide search. Unfortunately, it's difficult to formalize a process for getting the appropriate domain knowledge into the search algorithm so it can do some good. The successes to date have resulted from ad hoc improvement of programs that didn't work as well initially.

The field of efficient control of the multiagent discrete differential games needs new technology from the science of artificial intelligence. This technology, the so-called Linguistic Geometry, is being developed in [41-61].

There are many such problems where human expert skills in reasoning about complex goal-oriented systems are incomparably higher than the level of modern computing systems. Unfortunately, problems of tactics planning and automatic control of autonomous agents such as aerospace vehicles, space stations and robots with cooperative and opposing interests are of the type where human problem-solving skills can not be directly applied. Moreover, there are no highly-skilled human experts in these fields ready to substitute for robots (on a virtual model) or transfer their knowledge to them. There is no grand-master in robot control, although, of course, the knowledge of existing experts in this field should not be neglected – it is even more valuable. Due to the special significance of these problems and the fabulous costs of mistakes, the quality of solutions must be very high and usually subject to continuous improvement.

In this respect it is very important to study human expert reasoning about similar complex systems in the areas where the results are successful, in order to discover the keys to success, and then apply and adopt these keys to the new, as yet, unsolved problems, and first and foremost to the aerospace critical complex systems. It should be considered as investigation, development, and consequent expansion of advanced human expert skills into new areas.

2 Theoretical Background

The difficulties we encounter trying to find the optimal operation for real-world complex control systems are well known. While the formalization of the problem, as a rule, is not difficult, an algorithm that finds its solution usually results in the search of many variations. For small-dimensional "toy" problems a solution can be obtained; however, for most real-world problems the

dimension increases and the number of variations increases significantly, usually exponentially, as a function of dimension [5]. Thus, most real-world search problems are not solvable with the help of exact algorithms in a reasonable amount of time. This becomes increasingly critical for the real-time aerospace autonomous and semiautonomous vehicles and robots [2, 6].

There have been many attempts to find the optimal (suboptimal) operation for real-world complex systems, in particular, for aerospace applications [7-9]. Basically, all the approaches for the limited time search can be broken into four categories: the imprecise computation [10], real-time search [11], approximate processing [12], and anytime algorithms [13]. According to [6] the correct pruning in its many manifestations is still the only technique that reduces the worst-case execution time without compromising the goal state. But for real-world applications this reduction is usually insufficient: it does not overcome the combinatorial explosion. Another techniques, such as approximate processing, scoping, and use of domain knowledge, can reduce execution time significantly but they might compromise the goal state.

One of the basic ideas is to decrease the dimension of the real-world system following the approach of a *human expert in the field*, by breaking the system into smaller subsystems. This process of decomposition can be applied recursively until we end up with a collection of basic subproblems that can be treated (in some sense) independently. These ideas have been implemented for many problems with varying degrees of success (see, e.g., [14-17]). Implementations based on the formal theories of linear and nonlinear planning meet hard efficiency problems [18-22]. An efficient planner requires an intensive use of heuristic knowledge. Moreover it is possible to use both dynamic and static heuristic knowledge in reducing the search variations. The dynamic knowledge can be acquired during the run time and immediately applied for search reduction [6]. On the other hand, a pure heuristic implementation is unique. There is no general constructive approach to such implementations. Each new problem should be carefully studied, and previous experience usually can not be applied. Basically, we can not answer the question: what are the formal properties of the human expert heuristics that drove us to a successful hierarchy of subsystems for a given problem, and how can we apply the same ideas in a an altered or even different problem domain? Moreover, every attempt to evaluate the computational complexity and quality of a pilot solution necessitates implementing its program, which in itself is a unique task for each problem.

In the 1960's, a formal syntactic approach to the investigation of properties of natural language resulted in the fast development of a theory of formal languages by Chomsky [23], Ginsburg [24], and others. This development provided an interesting opportunity for dissemination of this approach to different areas. In particular, there came an idea of analogous linguistic representation of images. This idea was successfully developed into syntactic methods of pattern recognition by Fu [25], Narasimhan [26], and Pavlidis [27], and picture description languages by Shaw [28], Feder [29], and Rosenfeld [30].

Searching for adequate mathematical tools formalizing human heuristics of dynamic hierarchies, we have transformed the idea of linguistic representation of complex real-world and artificial images into the idea of similar representation of complex hierarchical systems [40]. However, the appropriate languages should possess more sophisticated attributes than languages usually used for pattern description. The origin of such languages can be traced back to the research on programmed attribute grammars by Knuth [31], Rozenkrantz [32], and Volchenkov [33].

A mathematical environment (a "glue") for the formal implementation of this approach was developed following the theories of formal problem solving and planning by Nilsson [20], Fikes and Nilsson [34], Sacerdoti [22], McCarthy [35], McCarthy and Hayes [36], and others based on first order predicate calculus.

In the beginning of 80's Botvinnik, Stilman, and others developed one of the most interesting and powerful heuristic hierarchical models. It was successfully applied to scheduling, planning, control, and computer chess. The hierarchical networks were introduced in [17, 37] in the form of ideas, plausible discussions, and program implementations. We consider this model as an ideal case for transferring the developed search heuristics to other domains employing formal linguistic tools.

An application of the developed model to a chess domain was implemented in full as program PIONEER [17]. Similar heuristic model was implemented for power equipment maintenance in a number of computer programs being used for maintenance scheduling all over the former USSR [38-40].

3. Heuristic Search Efficiency

Here we discuss the parameters of the search and the criteria for evaluation of results. Such parameters of the system as number of agents, size of the space for their motions, and length of the variant-solution, can be considered as characteristics of the complexity of this class of problems. For example, the length of a solution usually predetermines the depth of the search tree which is necessary to generate and evaluate. Thus, if a 6-move search is required, we have to generate a search tree of the depth 6. The question is: what is the "average breadth" of this tree, i.e., how many moves (on average) should be included into this tree at each node? For example, applying the brute force search algorithm, we have to include all the moves permitted in every state according to the problem statement. It means that in this case we have to generate a search tree of the size:

$$B+B^2+\ldots+B^L=T \qquad (1)$$

where B is the average number of moves in each state, L is the depth of the search, and T is the total number of states generated. Following Nilsson [20] parameter B is called a *branchingfactor*. The computation of B is based on the consideration of a hypothetical search tree with the depth of all branches equal to L, total number of moves equal to T , and a *constant* number of successors of each node. According to (1) this hypothetical constant number is equal to the branching factor B and might be computed as the solution of eq.(1) relative to B. Big values of B correspond to a non-selective search; obviously they indicate an exponential growth of the search (with a big base) as a function of the length of a solution. We look for approximate algorithms that reduce B, especially, those algorithms which make B close to 1. Such algorithms should be considered as extremely goal-driven with *minimal branching to different directions.*

4. Introduction to Linguistic Geometry

To discover the inner properties of human expert heuristics, which have been successful in a certain class of complex control systems, we develop a formal theory, the so-called *Linguistic Geometry* [41-56]. This research includes the development of syntactic tools for *knowledge representation* and *reasoning* about large-scale hierarchical complex systems. It relies on the formalization of *search heuristics*, which allow one to decompose complex system into a hierarchy of subsystems, and thus solve intractable problems by reducing the search. These *hierarchical images* in the form of networks of paths were extracted from the expert vision of the problem.

The hierarchy of subsystems is represented as a *hierarchy of formal attribute languages* where each "sentence" (a group of "words" or symbols) of the lower level language corresponds to the "word" of the higher level one. Following a linguistic approach each subsystem could be represented as a string of symbols with parameters: $\boldsymbol{a}(x_1)\boldsymbol{a}(x_2)\ldots\boldsymbol{a}(x_n)$, where the values of the parameters incorporate the semantics of the problem domain or lower-level subsystems. The lowest-level language of the hierarchy of languages, the Language of Trajectories [41,43,44, 46], serves as a building block to create the upper-level languages, the Languages of Networks [45-50].

The Language of Trajectories actually is a formalization of the description of the set of various paths between different points of the a complex control system. An element might follow a path to achieve the goal "connected with the ending point of this path." The Language of Networks is a formalization of a set of networks of certain paths unified by the mutual goal. For example, in the chess model such a network represents planning for a local fight, in the robot control model an analogous network of planning paths represents a draft short-range plan for approaching local goal in hazardous environment, i.e., getting over mobile and immobile obstacles. In the scheduling problem it corresponds to the maintenance schedule of a certain power unit including the schedule

for the provision of resources required.

Network languages allow us to describe the "statics", i.e., the states of the System. In order to describe the "dynamics" of the System, i.e., the motions from one state to another, we have to regenerate the entire hierarchy of languages. Of course, it is an inefficient procedure. To improve the efficiency of applications in the search process it is important to describe the change of the hierarchy of languages [50]. A study of this change helped us in modifying the hierarchy instead of regenerating it in each state. This change is represented as a mapping (translation) to some other hierarchy (actually, to the new state of the same hierarchy). Thus, the functioning of the system, in a search process, generates a tree of translations of the hierarchy of languages. This tree is represented as a string of the highest level formal language, the Language of Translations [50, 51, 53].

5 Class of Problems

A **Complex System** is the following eight-tuple:

$< X, P, R_p, \{ON\}, v, S_i, S_t, TR>$, where

$X=\{x_i\}$ is a finite set of *points*;

$P=\{p_i\}$ is a finite set of *elements*; P is a union of two non-intersecting subsets P_1 and P_2;

$R_p(x, y)$ is a set of binary relations of *reachability* in X (x and y are from X, p from P);

ON(p)=x, where ON is a partial function of *placement* from P into X;

v is a function on P with positive integer values describing the *values* of elements.

The Complex System searches the state space, which should have initial and target states;

S_i and S_t are the descriptions of the *initial* and *target* states in the language of the first order predicate calculus, which matches with each relation a certain Well-Formed Formula (WFF). Thus, each state from S_i or S_t is described by a certain set of WFF of the form $\{ON(p_j) = x_k\}$;

TR is a set of operators, TRANSITION(p, x, y), of transitions of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed from and added to the description of the state), and of WFF of applicability of the transition. Here,

Remove list: ON(p)=x, ON(q)=y;

Add list: ON(p)=y;

Applicability list: $(ON(p)=x)\wedge R_p(x,y)$,

where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out with participation of one *or many elements* p from P_1 and P_2.

According to the definition of the set P, the elements of the System are divided into two subsets P_1 and P_2. They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., $R_p(x, y)$ holds. The current location of each element is described by the equation ON(p)=x. Thus, the description of each state of the System $\{ON(p_j)=x_k\}$ is the set of descriptions of the locations of the elements. The operator TRANSITION(p, x, y) describes the change of the state of the System caused by the move of the element p from point x to point y. The element q from point y must be withdrawn (eliminated) if p and q do not belong to the same one of the two subsets P_1 and P_2.

The problem of the optimal operation of the System is considered as a search for the optimal sequence of transitions leading from one of the initial states of S_i to a target state S of S_t.

A new approach to definition of Complex Systems is considered in [59, 60].

It is easy to show formally that a robotic system can be considered as a Complex System (see below). Many different technical and human society systems (including military battlefield systems, systems of economic competition, positional games) that can be represented as twin sets of movable units (representing two or more opposing sides) and their locations can be considered

as Complex Systems.

With such a problem statement for the search of the optimal sequence of transitions leading to the target state, we could use formal methods like those in the problem-solving system STRIPS [34], nonlinear planner NOAH [22], or in subsequent planning systems. However, the search would have to be made in a space of a huge dimension (for nontrivial examples). Thus, in practice no solution would be obtained.

We devote ourselves to finding an approximate solution of a reformulated problem.

6 Geometry of Complex Systems: Measurement of Distances

To create and study a hierarchy of dynamic subsystems, we have to investigate geometrical properties of the Complex System.

A ***map of the set*** **X** relative to the point x and element p for the Complex System is the mapping: $\mathbf{MAP_{x,p}}$: X —> $\mathbf{Z_+}$ (where x is from X, p is from P), which is constructed as follows. We consider a *family of reachability areas* from the point x, i.e., a finite set of the following nonempty subsets $\{M^k_{x,p}\}$ of X (Fig.1)**:**

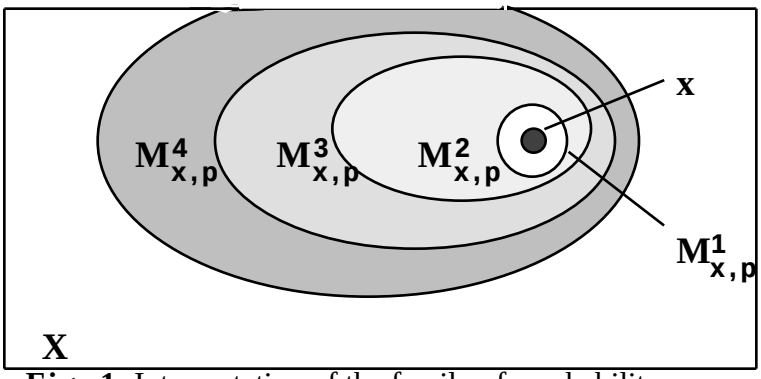

Fig. 1. Interpretation of the family of reachability areas

k=1: $M^k_{x,p}$ is a set of points *m reachable in one step* from x: $R_p(x,m)$=T;

k>1: $M^k_{x,p}$ is a set of points *reachable in k steps and not reachable in k*-1 *steps,* i.e., points *m* reachable from points of $M^{k-1}_{x,p}$ and not included in any $M^i_{x,p}$ with *i* less than *k*.

Let $MAP_{x,p}(y)=k$, for y from $M^k_{x,p}$ (*the number of steps from x to y*). For the remaining points, let $MAP_{x,p}(y)=2n$, if $y \neq x$ (n is the number of points in X)**;** $MAP_{x,p}(y)=0$, if y = x.

It is easy to verify that the map of the set X for the specified element p from P defines an *asymmetric distance function* on X:

1. $MAP_{x,p}(y) > 0$ for $x \neq y$; $MAP_{x,p}(x)=0$;

2. $MAP_{x,p}(y)+MAP_{y,p}(z) \geq MAP_{x,p}(z)$.

If R_p is a symmetric relation,

3. $MAP_{x,p}(y)=MAP_{y,p}(x)$.

In this case each of the elements p from P specifies on X its *own* ***metric*****.**

Examples of measurement of distances for robotic vehicles are considered in Section 10. Different examples are presented in [41, 46, 53].

7 Set of Paths: Language of Trajectories

This language is a formal description of the set of lowest-level subsystems, the set of all paths between points of the Complex System. An element might follow a path to achieve the goal

"connected with the ending point" of this path.

A ***trajectory*** for an element p of P with the beginning at x of X and the end at the y of X ($x \neq y$) with a length l is following formal string of symbols $\boldsymbol{a}(x)$ with points of X as parameters:

$$t_o = \boldsymbol{a}(x)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_l),$$

where $x_l = y$, each successive point x_{i+1} is reachable from the previous point x_i, i.e., $R_p(x_i, x_{i+1})$ holds for i = 0, 1,…, l–1; element p stands at the point x: ON(p)=x. We denote by $t_p(x, y, l)$ the set of all trajectories for element p, beginning at x, end at y, and with length l. $\boldsymbol{P}(t_o)=\{x, x_1, ..., x_l\}$ is the set of parameter values of the trajectory t_o. (To avoid confusion we should emphasize that $\boldsymbol{a}(x)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_l)$ is a formal record and does not mean anything else except what is given above.)

A ***shortest trajectory*** t of $t_p(x, y, l)$ is the trajectory of minimum length for the given beginning x, end y, and element p.

Properties of the Complex System permit us to define (in general form) and study formal grammars for generating the shortest trajectories. A general grammar and its application to generating 2D shortest trajectories for a aerospace robotic vehicles is presented in Section 11. Different examples are considered in [52-54].

Reasoning informally, an analogy can be set up: the shortest trajectory is analogous with a straight line segment connecting two points in a plane. An analogy to a k-element segmented line connecting these points is called an ***admissible trajectory of degree*** **k**, i.e., the trajectory that can be divided into k shortest trajectories. The admissible trajectories of degree 2 play a special role in many problems. As a rule, elements of the System should move along the shortest paths. In case of an obstacle, the element should move around this obstacle by tracing an intermediate point aside and going to and from this point to the end along the shortest trajectories. Thus, in this case, an element should move along an admissible trajectory of degree 2.

A ***Language of Trajectories*** $L_t^H(S)$ for the Complex System in a state S is the set of all the shortest and admissible (degree 2) trajectories of length less than H. Different properties of this language and generating grammars were investigated in [41, 43, 44].

8 Networks of Paths: Languages of Trajectory Networks

After defining the Language of Trajectories, we have new tools for the breakdown of our System into subsystems. According to the ideas presented in [17], these subsystems should be various types of trajectory networks, i.e., the sets of interconnected trajectories with one singled out and called the *main trajectory*. An example of such network is shown in Fig. 2.

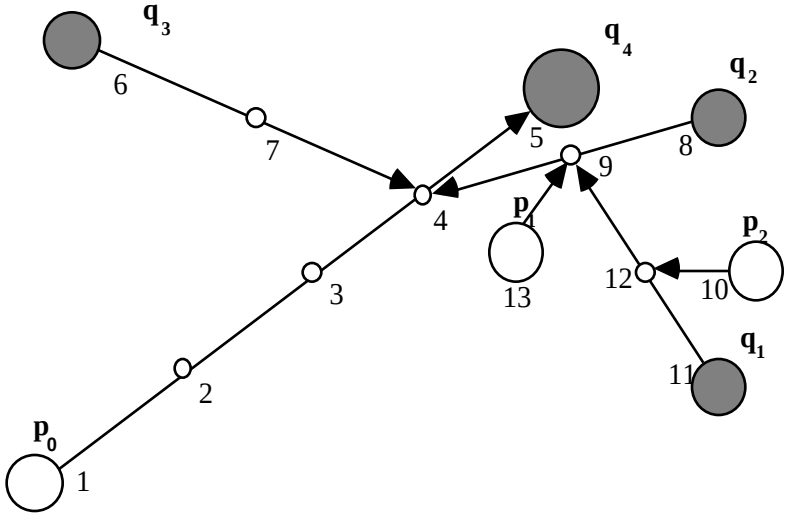

Fig. 2. Network language interpretation.

The basic idea behind these networks is as follows. Element p_o should move along the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ to reach the ending point 5 and remove the target q_4 (an opposing element). Naturally, the opposing elements should try to disturb those motions by controlling the intermediate points of the main trajectory. They should come closer to these points (to the point 4 in Fig. 2) and remove element p_o after its arrival (at point 4). For this purpose, elements q_3 or q_2 should move along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$, respectively, and wait (if necessary) on the next to last point (7 or 9) for the arrival of element p_o at point 4. Similarly, element p_1 of the same side as p_o might try to disturb the motion of q_2 by controlling point 9 along the trajectory $\boldsymbol{a}(13)\boldsymbol{a}(9)$. It makes sense for the opposing side to include the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of element q_1 to prevent this control.

Similar networks are used for the breakdown of complex systems in different areas. Let us consider a linguistic formalization of such networks. The Language of Trajectories describes "one-dimensional" objects by joining symbols into a string employing a reachability relation $R_p(x, y)$. To describe networks, i.e., “multi-dimensional" objects made up of trajectories, we use the relation of *trajectory connection.*

A ***trajectory connection*** of the trajectories t_1 and t_2 is the relation $C(t_1,t_2)$. It holds if the ending link of the trajectory t_1 coincides with an intermediate link of the trajectory t_2; more precisely, t_1 is connected with t_2 if among the parameter values $\boldsymbol{P}(t_2)=\{y,y_1,\ldots,y_l\}$ of trajectory t_2 there is a value $y_i = x_k$, where $t_1=\boldsymbol{a}(x_o)\boldsymbol{a}(x_1)\ldots\boldsymbol{a}(x_k)$. If t_1 belongs to a set of trajectories with the common end-point, then the entire set is said to be connected with the trajectory t_2.

For example, in Fig. 2 the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ are connected with the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ through point 4. Trajectories $\boldsymbol{a}(13)\boldsymbol{a}(9)$ and $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ are connected with $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$.

To formalize the trajectory networks, we define and use routine operations on the set of trajectories: $C_A^k(t_1,t_2)$, a ***k-th degree of connection***, and $C_A^+(t_1,t_2)$, a ***transitive closure***.

Trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ in Fig. 2 is connected degree 2 with trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$, i.e., $C^2(\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9),\ \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5))$ holds. Trajectory $\boldsymbol{a}(10)\boldsymbol{a}(12)$ in Fig. 2 is in transitive closure to the trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ because $C^3(\boldsymbol{a}(10)\boldsymbol{a}(12),\ \boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5))$ holds by means of the chain of trajectories $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$.

A ***trajectory network*** **W** relative to trajectory t_o is a finite set of trajectories $t_o,t_1,\ldots,t_k$ from the language $L_t^H(S)$ that possesses the following property: for every trajectory t_i from W (i = 1, 2,…,k) the relation $C_W^+(t_i,t_o)$ holds, i.e., each trajectory of the network W is connected with the trajectory t_o that was singled out by a subset of interconnected trajectories of this network. If the relation $C_W^{\mathbf{m}}(t_i, t_o)$ holds, i.e., this is the **m**-th degree of connection, trajectory t_i is called the **m** ***negation trajectory***.

Obviously, the trajectories in Fig. 2 form a trajectory network relative to the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$. We are now ready to define network languages.

A ***family of trajectory network languages*** $\mathbf{L_C(S)}$ in a state S of the Complex System is the family of languages that contains strings of the form

$$t(t_1, \textit{param})t(t_2, \textit{param})\ldots t(t_m, \textit{param}),$$

where *param* in parentheses substitute for the other parameters of a particular language. All the symbols of the string $t_1,\ t_2,\ldots,\ t_m$ correspond to trajectories that form a trajectory network W relative to t_1.

Different members of this family correspond to different types of trajectory network languages, which describe particular subsystems for solving search problems. One such language is the language that describes specific networks called Zones. They play the main role in the model considered here [17, 37, 45, 47, 50]. A formal definition of this language is essentially constructive and requires showing explicitly a method for generating this language, i.e., a certain formal grammar, which is presented in [45, 50]. In order to make our points transparent here, we

define the Language of Zones informally.

A ***Language of Zones*** is a trajectory network language with strings of the form

$$Z=t(p_o,t_o,\tau_o)\ t(p_1,t_1,\tau_1)\ldots t(p_k,t_k,\tau_k),$$

where $t_o,t_1,\ldots,t_k$ are the trajectories of elements $p_o,p_2,\ldots,p_k$ respectively; $\tau_o,\tau_1,\ldots,\tau_k$ are nonnegative integers that "denote the time allotted for the motion along the trajectories" in a correspondence to the mutual goal of this Zone: to remove the target element – for one side, and to protect it – for the opposing side. Trajectory $t(p_o,t_o,\tau_o)$ is called the *main trajectory* of the Zone. The element q standing on the ending point of the main trajectory is called the *target*. The elements p_o and q belong to the opposing sides.

To make it clearer, let us show the Zone corresponding to the trajectory network in Fig. 2.

$$Z=\boldsymbol{t}(p_o,\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5),4)\ \boldsymbol{t}(q_3,\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4),3)\boldsymbol{t}(q_2,\ \boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4),\ 3)\boldsymbol{t}(p_1,\ \boldsymbol{a}(13)\boldsymbol{a}(9),\ 1)$$
$$\boldsymbol{t}(q_1,\ \boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9),\ 3)\ \boldsymbol{t}(p_2,\ \boldsymbol{a}(10)\boldsymbol{a}(12),\ 1)$$

Assume that the goal of the white side is to remove target q_4, while the goal of the black side is to protect it. According to these goals, element p_o starts the motion to the target, while black starts in its turn to move elements q_2 or q_3 to intercept element p_o. Actually, only those black trajectories are to be included into the Zone where the motion of the element makes sense, i. e., the *length of the trajectory is less than the amount of time (third parameter* τ*) allocated to it.* For example, the motion along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$ makes sense, because they are of length 2 and time allocated equals 3: each of the elements has 3 time intervals to reach point 4 to intercept element p_o assuming one would go along the main trajectory without move omission and all the intercepting elements will move *simultaneously* (if necessary). According to definition of Zone, the trajectories of white elements (except p_o) could only be of the length 1, e.g., $\boldsymbol{a}(13)\boldsymbol{a}(9)$ or $\boldsymbol{a}(10)\boldsymbol{a}(12)$. As element p_1 can intercept the motion of the element q_2 at the point 9, black includes into the Zone the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of the element q_1, which has enough time for motion to prevent this interception. The total amount of time allocated to the whole bunch of black trajectories connected (directly or indirectly) with the given point of the main trajectory is determined by the number of that point. For example, for the point 4, it equals 3 time intervals.

A language $L_Z{}^H(S)$ generated by the certain grammar $\mathbf{G_Z}$ [45-48, 50] in a state S of a Complex System is called the ***Language of Zones***.

In the following Sections 9-19 we consider application of the Linguistic Geometry tools to the 2D optimization problems for 4 aircraft. The first problem reflects a search for an optimal combat scenario for the aircraft moving in a serial mode, i.e., one aircraft at a time and motions of the opposing sides alternate. Initially, this problem has been considered in [51]. The following problems are generalizations of the first one towards introduction of partial and total concurrency. This would allow some of the aircraft (Sections 13-16) or all of them (Sections 17-18) move simultaneously. *A concurrency* is an essential feature of the models of the real world combat scenarios and multiagent systems in general.

9 Robotic Air Combat as Complex System

The robotic model can be represented as a Complex System (Section 5) as follows. The set X represents the operational district, which could be the area of combat operation, broken into smaller square or cubic areas, "points", e.g., in the form of the big square or cubic grid. It could be a space operation, where X represents the set of different orbits, or an air force battlefield, etc. P is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests; $R_p(x,y)$ represent moving capabilities of different robots for different problem domains: robot p can move from point x to point y if $R_p(x, y)$ holds. Some of the robots can crawl, others

can jump or ride, sail and fly, or even move from one orbit to another. Some of them move fast and can reach point y (from x) in "one step", i.e., $R_p(x, y)$ holds, others can do that in *k* steps only, and many of them can not reach this point at all. ON(p)=x, if robot p is at the point x; v(p) is the value of robot p. This value might be determined by the technical parameters of the robot. It might include the immediate value of this robot for the given combat operation. S_i is an arbitrary initial state of operation for analysis, or the starting state; S_t is the set of target states. These might be the states where robots of each side reached specified points. On the other hand, S_t can specify states where opposing robots of the highest value are destroyed. The set of WFF $\{ON(p_j) = x_k\}$ corresponds to the list of robots with their coordinates in each state. TRANSITION(p, x, y) represents the move of the robot p from the location x to location y; if a robot of the opposing side stands on y, a removal occurs, i.e., robot on y is destroyed and removed.

10 Robotic Model with Serial Motions

Robots with various moving capabilities are shown in Fig. 3. The operational district X is the table 8 x 8. Robot W-FIGHTER (White Fighter) standing on h8, can move to any next square (shown by arrows). The other robot B-BOMBER from h5 can move only straight ahead, one square at a time, e.g., from h5 to h4, from h4 to h3, etc. Robot B-FIGHTER (Black Fighter) standing on a6, can move to any next square similarly to robot W-FIGHTER (shown by arrows). Robot W-BOMBER standing on c6 is analogous with the robot B-BOMBER; it can move only straight ahead but in reverse direction. Thus, robot W-FIGHTER on h8 can reach any of the points y {h7, g7, g8} in on step, i.e., $R_{W\text{-}FIGHTER}$(h8, y) holds, while W-BOMBER can reach only c8 in one step.

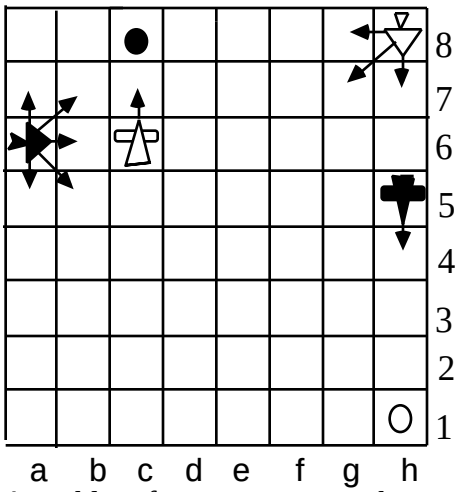

Fig. 3. A problem for autonomous robotic vehicles.

Assume that robots W-FIGHTER and W-BOMBER belong to one side, while B-FIGHTER and B-BOMBER belong to the opposite side: W-FIGHTER P_1, W-BOMBER P_1, B-FIGHTER P_2, B-BOMBER P_2. Also assume that two more robots, W-TARGET and B-TARGET, (unmoving devices or targeted areas) stand on h1 and c8, respectively. W-TARGET belongs to P_1, while B-TARGET P_2. Each of the BOMBERs can destroy unmoving TARGET ahead of the course; it also has powerful weapons capable to destroy opposing FIGHTERs on the next diagonal squares ahead of the course. For example W-BOMBER from c6 can destroy opposing FIGHTERs on b7 and d7. Each of the FIGHTERs is capable to destroy an opposing BOMBER approaching its location, but it also capable to protect its friendly BOMBER approaching its prospective location. In the latter case the joint protective power of the combined

weapons of the friendly BOMBER and FIGHTER can protect the BOMBER from interception. For example, W-FIGHTER located at d6 can protect W-BOMBER on c6 and c7.

The battlefield considered can be broken into two local operations. The first operation is as follows: robot B-BOMBER should reach point h1 to destroy the W-TARGET, while W-FIGHTER will try to intercept this motion. The second operation is similar: robot W-BOMBER should reach point c8 to destroy the B-TARGET, while B-FIGHTER will try to intercept this motion. After destroying the opposing TARGET the attacking side is considered as a winner of the local operation and the global battle. The only chance for the opposing side to avenge itself is to hit its TARGET on the next time interval and this way end the battle in a draw. The conditions considered above give us S_t, the description of target states of the Complex System. The description of the initial state S_i is obvious and follows from Fig. 3.

Assume that due to the shortage of resources (which is typical in real combat operation) or some other reasons, each side can not participate in both operations simultaneously. It means that during the current time interval, in case of White turn, either W-BOMBER or W-FIGHTER can move. Analogous condition holds for Blacks. Of course, it does not mean that if one side began participating in one of the operations it must complete it. Any time on its turn each side can switch from one operation to another, e.g., transferring resources (fuel, weapons, human resources, etc.), and later switch back. These constraints convert our problem into the air combat problem with ***serial*** motions. Later we will consider various modifications of these requirements including complete relaxation of them.

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-BOMBER from h5 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. Is there a strategy for the White side to make a draw? Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. Experiments with the computer programs showed that for a similar problem the search tree includes about a million moves (transitions). It is very interesting to observe the drastic reduction of search employing the Linguistic Geometry tools.

11 Generating Techniques

To demonstrate generation of the Hierarchy of Languages for this problem, below we consider generation of the Language of Trajectories for the robotic system on example of generation of the shortest trajectory from f6 to point h1 for the robot W-FIGHTER (Fig. 4). (This is the location of W-FIGHTER in one of the states of the System in the process of the search.)

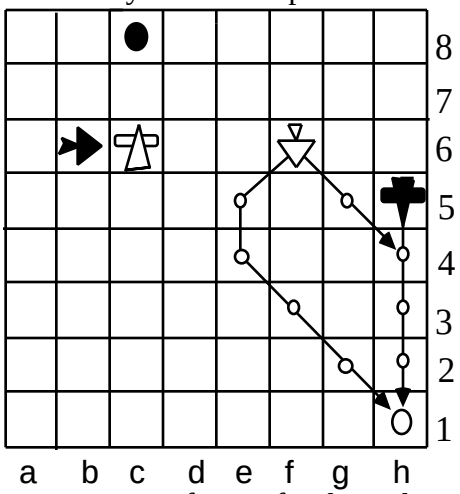

Fig. 4. Interpretation of Zone for the Robotic System.

Consider the Grammar of shortest trajectories $\mathbf{G_t^{(1)}}$ (Table I). This is a controlled grammar

[41]. Such grammars operate as follows.

Table I. A grammar of shortest trajectories $G_t^{(1)}$

L	Q	Kernel, π_k	F_T	F_F
1	Q_1	$S(x,y,l) \to A(x, y, l)$	two	ø
2_i	Q_2	$A(x,y,l) \to a(x)A(next_i(x,l),y,f(l))$	two	3
3	Q_3	$A(x, y, l) \to a(y)$	ø	ø

$V_T = \{a\}$ is the alphabet of terminal symbols,
$V_N = \{S, A\}$ is the alphabet of nonterminal symbols,
V_{PR} = *Truth Pred Con Var Func* {symbols of logical operations} is the alphabet of the first order predicate calculus ***PR***,
Truth$=\{T, F\}$
Pred $=\{Q_1, Q_2, Q_3\}$ are predicate symbols:
$Q_1(x, y, l) = (\mathrm{MAP}_{x,p}(y)=l)$ $(0<l<n)$
$Q_2(l) = (l \geq 1)$
$Q_3 = T$
Var $= \{x, y, l\}$ are variables;
Con $= \{x_o, y_o, l_o, p\}$ are constants;
Func = *Fcon* are functional symbols;
$Fcon=\{f, next_1, ..., next_n\}$ (n=|X|, number of points in X), $f(l)=l-1$, $D(f)=\mathbf{Z}_+\backslash\{0\}$
($next_i$ is defined lower)
$E = \mathbf{Z}_+ \cup X \cup P$ is the subject domain;
Parm: $S \to Var$, $A \to Var$, $a \to \{x\}$, is such a mapping that matches each symbol of the alphabet $V_T \cup V_N$ a set of formal parameters;
$L = \{1,3\} \cup$ two, two$=\{2_1, 2_2, ..., 2_n\}$ is a finite set called the set of labels; labels of different productions are different;
Q_i are the WFF of the predicate calculus ***PR***, the conditions of applicability of productions;
F_T is a subset of L of labels of the productions permitted on the next step derivation if Q=T; it is called a permissible set;
F_F is analogous to F_T but these productions are permitted in case of Q=F.
At the beginning of derivation: $x=x_o$, $y=y_o$, $l=l_o$, x_o X, y_o X, l_o $\mathbf{Z}_+$, p P.
$next_i$ is defined as follows:
$D(next_i) = X \times \mathbf{Z}_+ \times X^2 \times \mathbf{Z}_+ \times P$ (This is the domain of *next*)
$SUM=\{v \mid v$ X, $\mathrm{MAP}_{x_o,p}(v)+\mathrm{MAP}_{y_o,p}(v)=l_o\}$
$ST_k(x)=\{v \mid v$ from X, $\mathrm{MAP}_{x,p}(v)=k\}$,
$\mathrm{MOVE}_l(x)$ is an intersection of the following sets:
$ST_1(x)$, $ST_{l_o-l+1}(x_o)$ and SUM.
If $\mathrm{MOVE}_l(x)=\{m_1, m_2, ..., m_r\} \neq \emptyset$
then
$next_i(x, l)=m_i$ for $i \leq r$;
$next_i(x, l)=m_r$ for $r<i \leq n$,
otherwise
$next_i(x,l)=x$.

The initial permissible set of productions consists of the production with label 1. It should be

applied first. Let us describe the application of a production in such grammar. Suppose that we attempt to apply production with label l from L to rewrite a symbol A. We choose the leftmost entry of symbol A in the current string and compute the value of predicate Q, the condition of applicability of the production. If the current string *does not* contain A or $Q = F$, then the application of the production is ended, and the next production is chosen from the failure section F_F ; F_F becomes the current permissible set. If the current string *does* contain the symbol A and $Q=T$, A is replaced by the string in the right side of the production; we carry out the computation of the values of all formulas either standing separately (section π_n) or corresponding to the parameters of the symbols (π_k), and the parameters assume new values thus computed. Then, application of the production is ended, and the next production is chosen from the success section F_T, which is now the current permissible set. If the applicable section is empty, the derivation halts.

The controlled grammar shown in Table I can be used for generation of shortest trajectories for robots with arbitrary moving capabilities.

Let us consider W-FIGHTER from f6 (Fig. 4). Values of $MAP_{F6,W\text{-}FIGHTER}$ are shown in Fig. 5 (see also Section 6).

5	4	3	2	2	2	2	2
5	4	3	2	1	1	1	2
5	4	3	2	1	0	1	2
5	4	3	2	1	1	1	2
5	4	3	2	2	2	2	2
5	4	3	3	3	3	3	3
5	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5

Fig. 5. $MAP_{f6,FIGHTER}$

7	7	7	7	6	7	7	7
7	6	6	6	6	6	6	6
7	6	5	5	5	5	5	5
7	6	5	4	4	4	4	4
7	6	5	4	3	3	3	3
7	6	5	4	3	2	2	2
7	6	5	4	3	2	1	1
7	6	5	4	3	2	1	0

Fig. 6. $MAP_{h1,W\text{-}FIGHTER}$

Thus, the distance from f6 to h1 for W-FIGHTER is equal to 5. Applying the grammar $G_t^{(1)}$ we have (symbol $\stackrel{l}{=>}$ means application of the production with the label l):

$$S(f6, h1, 5) \stackrel{1}{=>} A(f6, h1, 5) \stackrel{21}{=>} a(f6)A(next_1(f6, 5), h1, 5)$$

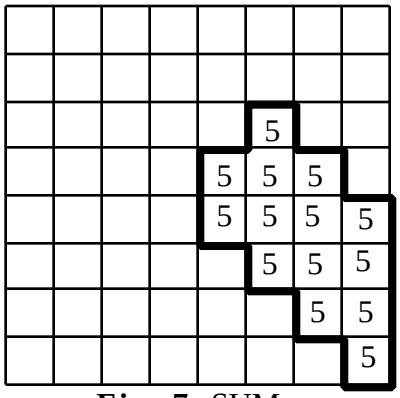

Fig. 7. SUM.

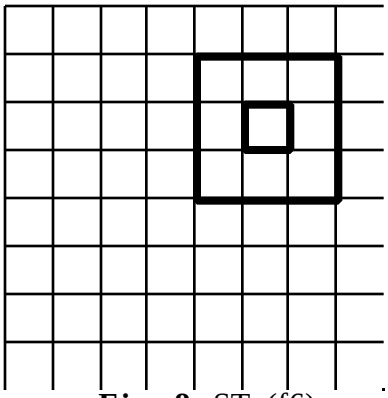

Fig. 8. $ST_1(f6)$.

Thus we have to compute MOVE (see definition of the function $next_i$ from the grammar $G_t^{(1)}$). First we have to determine the set of SUM, that is, we need to know values of $MAP_{f6,W\text{-}FIGHTER}$ and $MAP_{h1,W\text{-}FIGHTER}$ (shown in Fig. 6) on X. Adding these tables (Fig. 5 and

Fig. 6) as matrices we compute

$$SUM = \{v \mid v \in X,\ MAP_{f6,\,W\text{-}FIGHTER}(v) + MAP_{h1,W\text{-}FIGHTER}(v) = 5\}\ \text{(Fig. 7)}.$$

The next step is the computation of $ST_1(f6)=\{v \mid v \text{ from } X,\ MAP_{f6,W\text{-}FIGHTER}(v)=1\}$ which can be found in Fig. 8. In order to complete computation of the set $MOVE_5(f6)$ we have to determine the following intersection: $ST_1(f6)$, $ST_{5\text{-}5+1}(f6)=ST_1(f6)$ and SUM. Consequently, $MOVE_5(f6)=\{e5, f5, g5\}$; and $next_1(f6, 5)=e5$, $next_2(f6, 5)=f5$, $next_3(f6, 5)=g5$. Since the number of different values of *next* is equal to 3 (here r=3, see definition of the function *next*, Table I) we could branch at this step, apply productions 2_1, 2_2 and 2_3 simultaneously, and continue both derivations independently. This could be accomplished in a parallel computing environment. Let us proceed with the first derivation.

$$\boldsymbol{a}(f6)\boldsymbol{A}(e5,h1,4) \stackrel{2}{\Rightarrow}_1 \boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{A}(next_1(e5,4), h1,3)$$

We have to compute $next_1(e5, 4)$ and, as on the preceding step, have to determine $MOVE_4(e5)$. To do this we have to compute

$$ST_1(e5)=\{v \mid v \in X,\ MAP_{e5,W\text{-}FIGHTER}(v)=1\},\text{(Fig. 9)}$$

$$ST_{5\text{-}4+1}(f6) = ST_2(f6) = \{v \mid v \in X,\ MAP_{f6,W\text{-}FIGHTER}(v)=2\},\ \text{(Fig. 10)}.$$

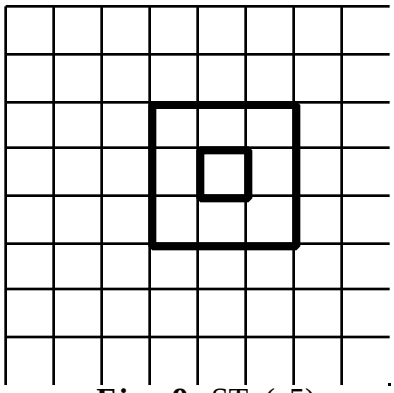

Fig. 9. $ST_1(e5)$

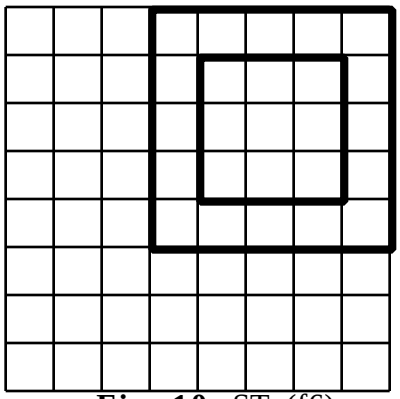

Fig. 10. $ST_2(f6)$.

The set of SUM is the same on all steps of the derivation. Hence, $MOVE_4(e5)$ is the intersection of the sets shown in Fig. 7, 9, 10; $MOVE_4(e5)= \{e4, f4\}$; and $next_1(e5,4) = e4$; $next_2(e5, 4) = f4$. Thus, the number of different values of the function *next* is equal to 2 (r=2), so the number of continuations of derivation should be multiplied by 2.

Let us proceed with the first one: $\boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{A}(e4, h1, 3) \stackrel{2}{\Rightarrow}_1 \ldots$ Eventually, we will generate one of the shortest trajectories for the robot W-FIGHTER from f6 to h1: $\boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{a}(e4)\boldsymbol{a}(f3)\boldsymbol{a}(g2)\boldsymbol{a}(h1)$.

Similar generating techniques are used to generate higher level subsystems, the networks of paths, i.e., the Language of Zones. For example one of the Zones to be generated in the state shown in Fig. 4 is as follows:

$$\boldsymbol{t}(\text{B-BOMBER},t_B,5)\boldsymbol{t}(\text{W-FIGHTER},t_F,5)\boldsymbol{t}(\text{W-FIGHTER},t_F^1,2), \text{ where}$$

$t_B = \boldsymbol{a}(h5)\boldsymbol{a}(h4)\boldsymbol{a}(h3)\boldsymbol{a}(h2)\boldsymbol{a}(h1)$, $t_F=\boldsymbol{a}(f6)\boldsymbol{a}(e5)\boldsymbol{a}(e4)\boldsymbol{a}(f3)\boldsymbol{a}(g2)\boldsymbol{a}(h1)$, $t_F^1=\boldsymbol{a}(f6)\boldsymbol{a}(g5)\boldsymbol{a}(h4)$

The details of generation of different Zones are considered in [45, 46].

12 Search Generation: Robotic Model with Serial Motions

Consider how the hierarchy of languages works for the optimal control of the Robotic System introduced above (Fig. 3). We generate the search of the Language of Translations representing it as a conventional search tree (Fig. 11) and comment on its generation.

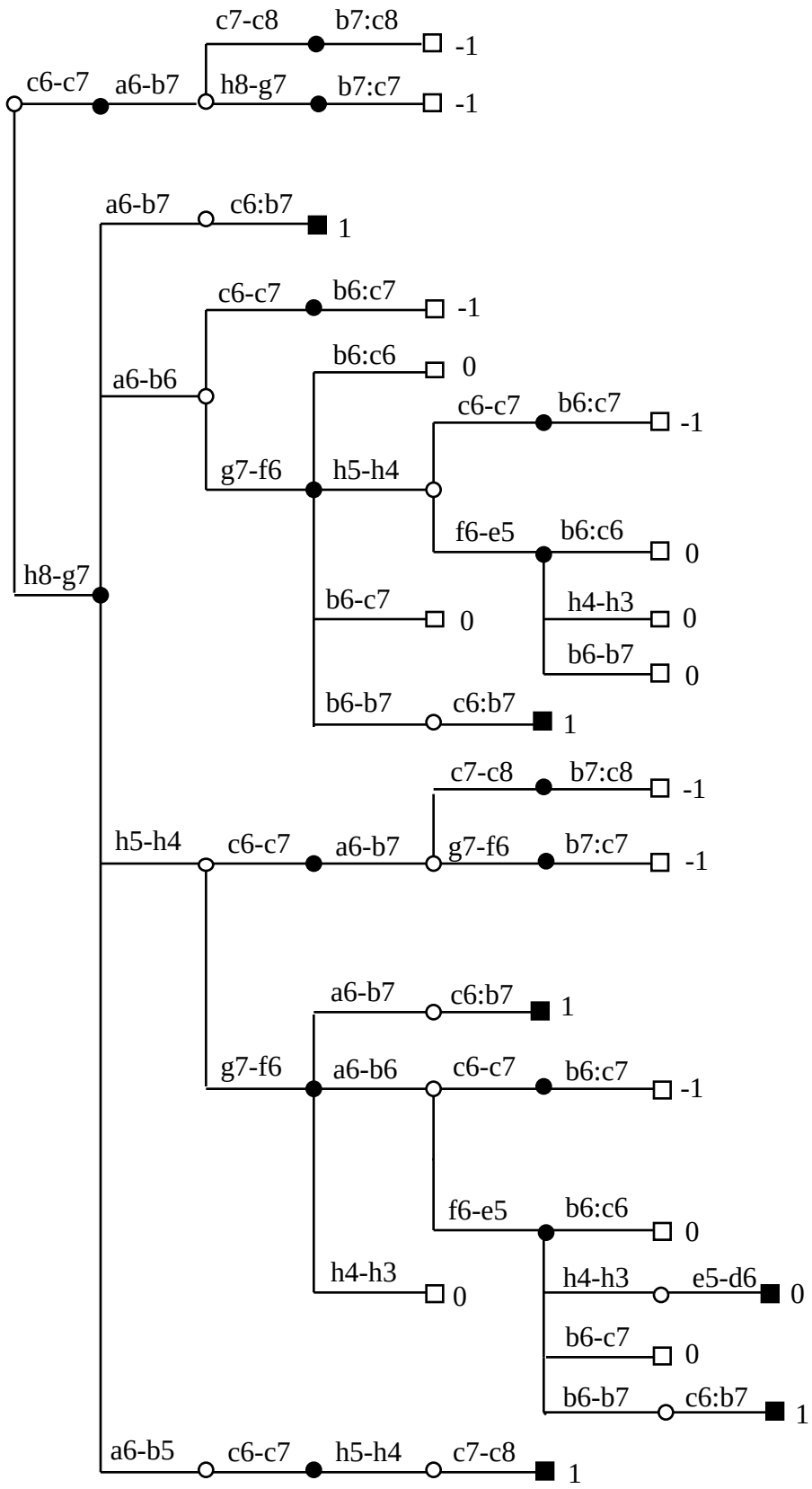

Fig. 11. Search tree for the optimization problem for robotic vehicles with serial motions.

First, the Language of Zones in the start state is generated. The targets for attack are determined within the limit of five steps. It means that horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of main trajectories of all Zones must not exceed 5 steps. The algorithm for choosing the right value of the horizon is considered in [56]. All the Zones generated in the start state are shown in Fig. 12. Zones for FIGHTERs as attacking elements are shown in the left diagram, while Zones for BOMBERs – in the right one. For example, one of the Zones for W-BOMBER, Z_{WB} is as follows:

Z_{WB}=***t***(P, ***a***(c6)***a***(c7)***a***(c8), 2)***t***(K, ***a***(a6)***a***(b7)***a***(c8), 3)***t***(K,***a***(a6)***a***(b7)***a***(c7), 2)***t***(P, ***a***(c6)***a***(b7), 1)

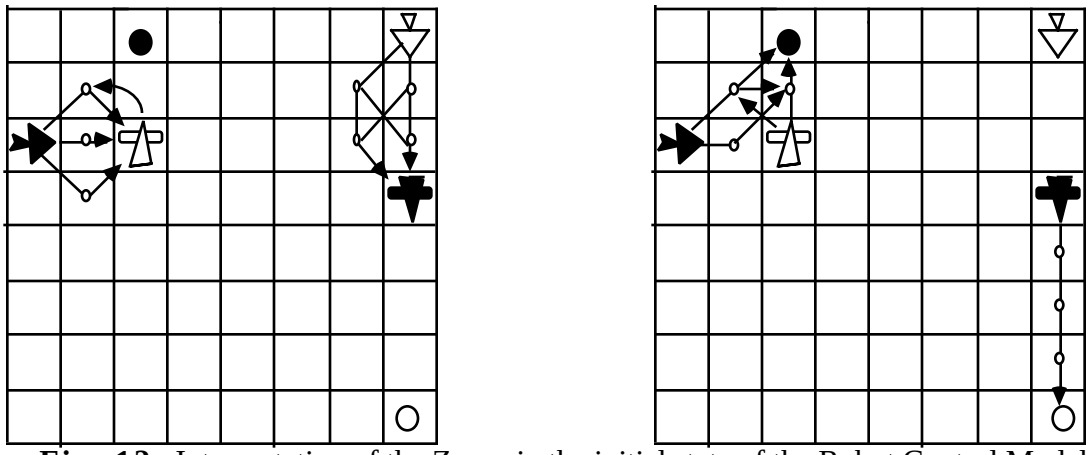

Fig. 12. Interpretation of the Zones in the initial state of the Robot Control Model.

The second trajectory of B-FIGHTER ***a***(a6)***a***(b6)***a***(c7) leading to the square c7 is included into different Zone; for each Zone only one trajectory from each bundle of trajectories is taken.

Generation begins with the move 1. c6-c7 in the White Zone with the target of the highest value and with the shortest main trajectory. The order of consideration of Zones and particular trajectories is determined by the grammar of translations. The computation of move-ordering constraints is the most sophisticated procedure in this grammar. It takes into account different parameters of Zones, trajectories, and the so-called chains of trajectories.

Next move, 1. ... a6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8 b7:c8 (Fig. 13, left). Symbol ":" means the removal of element. Here the grammar cuts this branch with the value of -1 (as a win of the Black side). This value is given by the special procedure of "generalized square rules" built into the grammar.

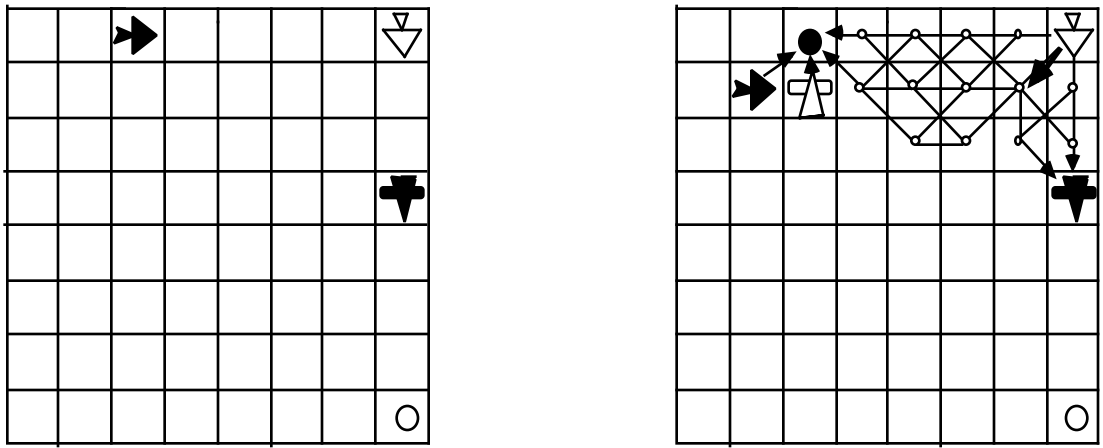

Fig. 13. States where the control Zone from h8 to c8 was detected (left) and where it was included into the search (right)

Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search. After climb up to the move 1. ... a6-b7, the tree to be analyzed consists of one branch (of two plies): 2. c7-c8 b7:c8. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange on c8 (in favor of the White side). This can be achieved by participation of W-FIGHTER from h8, i.e., by generation and inclusion of the new so-called "control" Zone with the main trajectory from h8 to c8. The set of different Zones from h8 to c8 (the bundle of Zones) is shown in Fig. 13, right. The move-ordering procedure picks the subset of Zones with main trajectories passing g7. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing W-BOMBER on h5. The motion along such trajectories allows to "gain time", i.e., to approach two goals simultaneously.

The generation continues: 2. h8-g7 b7:c7. Again, the procedure of "square rules" cuts the branch, evaluates it as a win of the black side, and the grammar initiates the climb. Analogously to the previous case, the inspection procedure determined that the current minimax value (-1) can be improved by the improvement of the exchange on c7. Again, this can be achieved by the inclusion of Zone from h8 to c7. Of course, the best "time-gaining" move in this Zone is 2. h8-g7, but it was already included (as move in the Zone from h8 to c8), and it appeared to be useless. No other branching at this state is generated.

The inspection procedure does not find new Zones to improve the current minimax value, and the climb continues up to the start state. The analysis of the subtree shows that inclusion of Zone from h8 to c8 in the start state can be useful: the minimax value can be improved. Similarly, the most promising "time-gaining" move is 1. h8-g7. The Black side responded 1. ... a6-b7 along the first negation trajectories ***a***(a6)***a***(b6)***a***(c7) and ***a***(a6)***a***(b6)***a***(c8) (Fig. 12, right). Obviously, 2. c6:b7, and the branch is terminated. The grammar initiates the climb and move 1. ... a6-b7 is changed for 1. ... a6-b6 along the trajectory ***a***(a6)***a***(b6)***a***(c7). Note, that grammar "knows" that in this state trajectory ***a***(a6)***a***(b6)***a***(c7) is active, i.e., B-FIGHTER has enough time for interception. The following moves are in the same Zone of W-BOMBER: 2. c6-c7 b6:c7. This state is shown in Fig. 14, left. The "square rule procedure" cuts this branch and evaluates it as a win of the Black side.

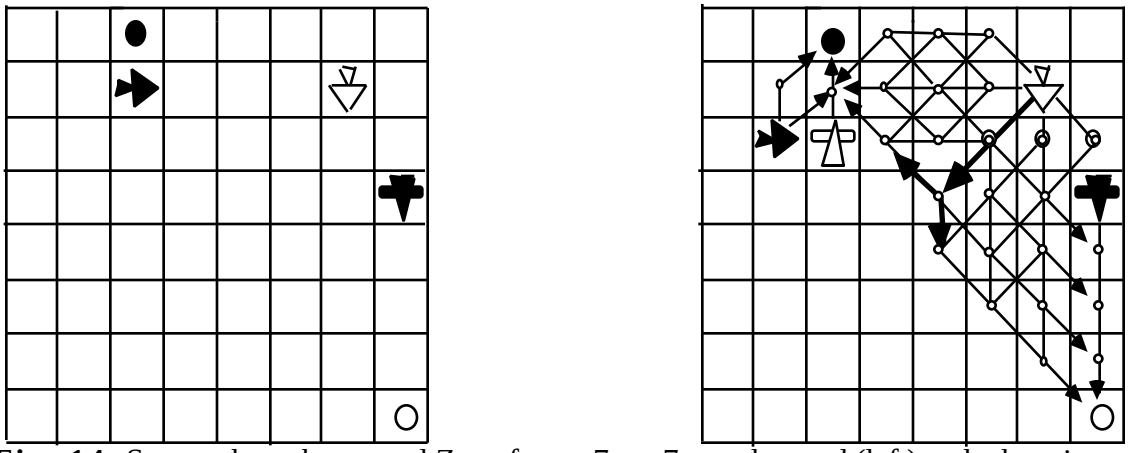

Fig. 14. States where the control Zone from g7 to c7 was detected (left) and where it was included into the search (right).

New climb up to the move 2. ... a6-b6 and execution of the inspection procedure resulted in the inclusion of the new control Zone from g7 to c7 in order to improve the exchange on c7. The set of Zones with different main trajectories from g7 to c7 is shown in Fig. 14, right. Besides that, the trajectories from g7 to h4, h3, h2, and h1 are shown in the same Fig. 14. These are "potential" first negation trajectories. It means that beginning with the second symbol ***a***(f6),***a***(g6) or ***a***(h6) these trajectories become *first negation* trajectories (Section 8) in the Zone of B-BOMBER h5. Speaking informally, from squares f6, g6, and h6 W-FIGHTER can intercept B-BOMBER (in case of white move). The move-ordering procedure picks the subset of Zones with the main

trajectories passing f6. These trajectories partly coincide with the potential first negation trajectories. The motion along such trajectories allows to gain time, i.e., to approach two goals simultaneously. Thus, 2. g7-f6.

This way proceeding with the search we will generate the tree that consists of 49 moves. Obviously, this is a dramatic reduction in comparison with a million-move trees generated by conventional search procedures. The maximum depth of this search is 7, so from eq.(1) (Section 3) we find that the branching factor is about 1.5. This means that the search is highly goal-oriented.

13 Serial Mode Relaxation: Partial Concurrency

In this Section we are going to relax the most restrictive requirement of participation of the only element in each motion which holds for the robotic model considered above (Section 10) and in our earlier papers for the similar problems [49-56]. This relaxation is very important for the transfer of the Linguistic Geometry tools to real world problems. It was inherited from the original domain, the board games, which served as a testbed for the development of the Linguistic Geometry tools [17, 37]. At the same time we should emphasize that this requirement has never been a part of the general definition of Complex System (Section 5), though no examples of Complex Systems with simultaneous motions have ever been introduced. In the air combat problem considered above (Sections 10-12) this requirement was imposed in the form of the restriction for each side to participate in both local operations simultaneously.

Assume that in the air combat problem (Fig. 3) motions of the opposing sides alternate and each side can participate in both operations ***simultaneously***. It means, for example, that during the current time interval, in case of White turn, both W-BOMBER and W-FIGHTER, one of them, or none of them can move. Analogous condition holds for Black.

Despite the impression that local operations are independent because they are located far from each other, we know from the above that, actually, these operations are connected by “time sharing”. Moreover, despite the impression that local operation of B-BOMBER from h6 is an unconditionally winning operation, and the B-BOMBER operation looks like a guaranteed loss, there is a strategy for W-FIGHTER to use the time sharing and finish the global battle in a draw. After allowing concurrent motions for both sides, the question remains: is there a strategy for the White side to make a draw?

Analogously with the serial model, this question can be answered by the direct search employing, for example, minimax algorithm. It would result in a search tree of $18^4 \approx 105{,}000$ moves (transitions) taking into account that 4 plies is the minimum depth required to solve this problem. Let us observe the reduction of search employing the Linguistic Geometry tools.

To demonstrate generation of the Hierarchy of Languages for this problem, we have to generate the Language of Trajectories and the Language of Zones in each state of the search. In case of concurrent motions, the languages and grammars of Trajectories and Zones remain the same as for serial case because the single move requirement (as it was already mentioned) has never been a part of the definition of Complex System.

However, in this case we can introduce a new language, the Language of Concurrent Zones, where time allotted for the motion along the trajectories of higher negations is exactly the same as for the first negation. This will be a topic of our future papers. For the given problem this introduction is unimportant and we will use the conventional Language of Zones. The details of trajectories and Zones generation are considered in [41,45,46, 50].

14 Search Generation: Model with Partial Concurrency

Consider how the hierarchy of languages works for the optimal control of Robotic System with concurrent motions introduced above. We generate the string of the Language of Translations

representing it as a conventional search tree (Fig. 15) and comment on its generation. In our comments on this generation we will emphasize only major steps.

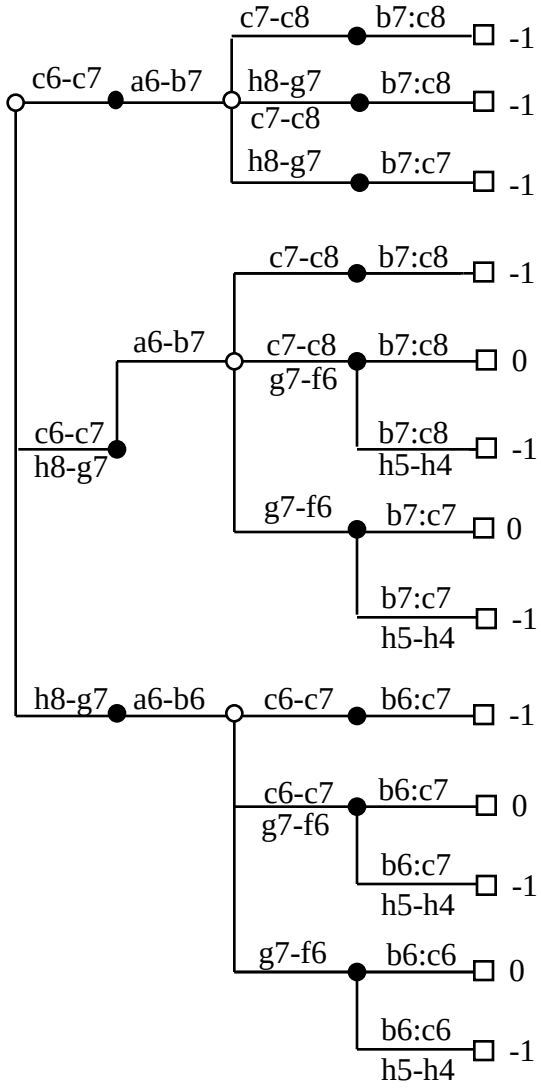

Fig. 15. Search tree for the 2D optimization problem for robotic vehicles with partial concurrency.

First, the Language of Zones in the start state is generated. Similar to the serial model and due to the same reasons, the targets for attack are determined within the limit of five steps. It means that the horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of the main trajectories of all Zones must not exceed 5 steps. All the Zones generated in the start state within the horizon 5 are exactly the same as in the serial case. They are shown in Fig. 12.

Generation begins with the move 1. c6-c7 in the White Zone with the target of the highest value and the shortest main trajectory ***a***(c6)***a***(c7)***a***(c8). The order of consideration of Zones and particular trajectories is determined by the Grammar of Translations. Next move, 1. ... a6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8 b7:c8 (Fig. 13, left). Here the grammar terminates this branch with the value -1 (as a win of the Black side). This value is given by the procedure of "generalized square rules" built into the grammar. This procedure determined that the only possible interceptor, W-FIGHTER, is out of the Zone of B-BOMBER, thus, it can not intercept B-BOMBER.

Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the

inspection procedure, the analysis of the subtree generated in the process of the earlier search. After climb up to the move 1. ... a6-b7, the tree to be analyzed consists of one branch (of two plies): 2. c7-c8 b7:c8. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange on c8 (in favor of the White side). This can be achieved by participation of W-FIGHTER from h8, i.e., by generation and inclusion of the new control Zones with the main trajectory from h8 to c8. These Zones have been detected (within the horizon 5) in the terminal state after the move 2. ... b7:c8 (Fig. 13, left). These Zones have been stored for possible activation at the higher levels of the search tree. The set of different Zones from h8 to c8 (the bundle of Zones) is shown in Fig. 13, right. The move-ordering procedure picks the subset of Zones with main trajectories passing g7. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing W-BOMBER on h5. The motion along such trajectories allows to "gain time", i.e., to approach two goals simultaneously.

The generation continues with the simultaneous motion of two agents, the double move, W-FIGHTER and W-BOMBER in their respective Zones: 2. h8-g7/c7-c8. The B-FIGHTER intercepts W-BOMBER at c8: 2. ... b7:c8. The "square rule" procedure terminated branch, evaluated it as a win (-1) for the Black side, and initiated backtracking climb. Move 2. h8-g7/c7-c8 is changed for the single move 2. h8-g7 with the similar response 2. ... b7:c7. The control Zones of W-BOMBER from h8 to c7 were detected and stored as idle. Analogously to the previous branch this branch was terminated, evaluated as -1, and the following backtracking climb stopped only at the initial state.

The grammar did not include different motions after 1. c6-c7 a6-b7, like 2. h8-g8, concluding that even the time-gaining motion along the shortest trajectory of the Zone from h8 to c8 was late, and, thus, useless. Now the Zone of W-FIGHTER from h8 to c8 can be activated at the top level of the search tree together with the attack Zone of W-BOMBER: 1. c6-c7/h8-g7. The following branch is exactly the same as the very first branch of the search tree generated so far: 1. ... a6-b7 2. c7-c8 b7:c8. The "square rule" procedure terminated branch, evaluated it as a win (-1) for the Black side, and initiated the backtracking climb.

New climb up to the move 2. ... a6-b7 and execution of the inspection procedure resulted in the inclusion of the groups of control Zones from g7 to c7 and to c8 in order to improve the exchanges at these locations. Both groups of Zones (to c7 and c8) have been detected earlier in the search tree. The set of Zones with different main trajectories from g7 to c7 and from g7 to c8 is shown in Fig. 14, right. Besides that, the trajectories from g7 to h4, h3, h2, and h1 are shown in the same Fig. 14. These are "potential" first negation trajectories. It means that beginning with the second symbol ***a***(f6), ***a***(g6) or ***a***(h6) these trajectories become first negation trajectories in the Zone of B-BOMBER on h5. Speaking informally, from squares f6, g6, and h6, Zone gateways, W-FIGHTER can intercept B-BOMBER (in case of White turn). The move-ordering procedure picks the subset of Zones with the main trajectories passing f6. These trajectories partly coincide with the potential first negation trajectories. The motion along such trajectories allows to "gain time", i.e., to approach two goals simultaneously. Thus, the double move 2. c7-c8/g7-f6 is included. After the response 2. ... b7:c8, the branch was terminated and evaluated as a draw (0) because the "square rule" procedure detected that W-FIGHTER is in the Zone of B-BOMBER and thus has enough time for interception. After the climb and branching, the attack Zone of B-BOMBER from h5 to h1 was activated. This Zone was detected in the initial state (Fig. 12, right) but has not been activated yet. The double move 2. ... b7:c8/h5-h4 resulted in termination of the branch with the value -1. The following climb and branching with inclusion of 2. g7-g6 as a single move resulted in 2. ... b7:c7/h5-h4 with terminal value -1.

The following generation shows that the minimax value of this search is -1, i.e., it is in favor of Black. It means that the allowance of concurrent motions prohibited White from the draw in this combat. The problem became almost trivial in this case. The search tree contains 28 moves while its depth is 4 moves. The branching factor is 1.96. It is easy to show that the average number of legal moves in each state of this problem is 18 which means that unreduced search tree should contain $18^4 \approx 105{,}000$ moves.

It would be reasonable to consider a *change* of this problem looking for a draw. Of course, we

should keep the allowance of the concurrent motions. This search for a change of the statement of the problem is quite typical for the real world problems. For example, let us assume that the combat modeling shows that our side would lose this combat. Then we would try to change the initial conditions, i.e., to restate the problem in such a way where our side can win or, at least, finish this combat in a draw. This would result in the change of the initial settlement of agents and resources. This is especially important if, firstly, a model gives guidelines for such a change, and, secondly, the check of the new settlement can be accomplished in real time. A new combat planning problem with concurrent motions is considered below. Initially, this model was introduced in [61].

15 Second Model with Partial Concurrency

Robots with different moving capabilities are shown in Fig. 16. The operational district X is the table 8 x 8. Robot W-FIGHTER (White Fighter) standing on h8, can move to any next square (shown by arrows). The other robot B-BOMBER (Black Bomber) from h6 can move only straight ahead, one square at a time, e.g., from h6 to h5, from h5 to h4, etc. Robot B-FIGHTER (Black Fighter) standing on a6, can move to any next square similarly to robot W-FIGHTER (shown by arrows). Robot W-BOMBER (White Bomber) standing on c6 is analogous with the robot B-BOMBER; it can move only straight ahead but in reverse direction. Thus, robot W-FIGHTER on h8 can reach any of the points y {h7, g7, g8} in one step, i.e., $R_{W\text{-}FIGHTER}$(h8, y) holds, while W-BOMBER can reach only c5 in one step.

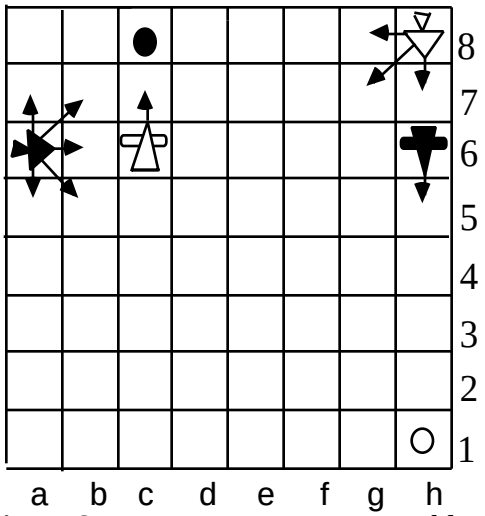

Fig. 16. New 2D optimization problem for robotic vehicles with partial concurrency.

Assume that robots W-FIGHTER and W-BOMBER belong to one side, while B-FIGHTER and B-BOMBER belong to the opposing side: W-FIGHTER P_1, W-BOMBER P_1, B-FIGHTER P_2, B-BOMBER P_2. Also assume that two more robots, W-TARGET and B-TARGET, (unmoving devices or targeted areas) stand on h1 and c8, respectively. W-TARGET belongs to P_1, while B-TARGET P_2. Each of the BOMBERs can destroy unmoving TARGET ahead of the course; it also has powerful weapons able to destroy opposing FIGHTERs on the next diagonal squares ahead of the course. For example, W-BOMBER from c6 can destroy opposing FIGHTERs on b7 and d7. Each of the FIGHTERs is able to destroy an opposing BOMBER approaching its location, but it also able to protect its friendly BOMBER approaching its prospective location. In the latter case the joint protective power of the combined weapons of the

friendly BOMBER and FIGHTER can protect the BOMBER from interception. For example, W-FIGHTER located at d6 can protect W-BOMBER on c6 and c7.

Analogously to the serial case, the battlefield considered can be broken into two local operations. The first operation is as follows: robot B-BOMBER should reach point h1 to destroy the W-TARGET, while W-FIGHTER will try to intercept this motion. The second operation is similar: robot W-BOMBER should reach point c8 to destroy the B-TARGET, while B-FIGHTER will try to intercept this motion. After destroying the opposing TARGET the attacking side is considered as a winner of the local operation and the global combat. The only chance for the opposing side to avenge is to hit its TARGET on the next time interval and this way end the battle in a draw. The conditions considered above give us S_t, the description of target states of the Complex System. The description of the initial state S_i is obvious and follows from Fig. 16.

Assume that motions of the opposing sides alternate and each side can participate in both operations ***simultaneously***. It means, for example, that during the current time interval, in case of White turn, both W-BOMBER and W-FIGHTER, one of them, or none of them can move. Analogous condition holds for Black. There is one exception. If W-FIGHTER hits B-BOMBER while the latter is fully armed, i.e., it is not at its final destination – square h1, W-BOMBER can not move simultaneously during this time interval in order to avoid possible consequences of the B-BOMBER explosion. Similar restriction holds for B-BOMBER: it can not move at the moment when W-BOMBER is destroyed (not at c8).

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-BOMBER from h6 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. The question is: is there a strategy for the White side to make a draw?

Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. Theoretical evaluations [20] of the lower bounds of complexity of this algorithm for the air combat problem show that, at best, it would result in a search tree of 25 million moves (transitions). In practice, even this number is unreachable. It is very interesting to observe the reduction of search employing the Linguistic Geometry tools.

To demonstrate generation of the Hierarchy of Languages for this problem, we have to generate the Language of Trajectories and the Language of Zones in each state of the search.

16 Search Generation: Second Model with Partial Concurrency

Consider how the hierarchy of languages works for the optimal control of the Robotic System introduced above (Fig. 16). We generate a string of the Language of Translations [50] representing it as a conventional search tree (Fig. 17) and comment on its generation. In fact, this tree is close to the search tree of the restricted problem with serial motions (Sections 10-12). In our comments on this generation we will emphasize the major steps.

First, the Language of Zones in the start state is generated. The targets for attack are determined within the limit of five steps. It means that horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of main trajectories of all Zones must not exceed 5 steps. The reasons and the algorithm for choosing the right value of the horizon are considered in [53, 54]. All the Zones generated in the start state within the horizon 5 are shown in Fig. 18. Zones for FIGHTERs as attacking elements are shown in the left diagram, while Zones for BOMBERs – in the right one.

Generation begins with the move 1. c6-c7 in the “white” Zone with the target of the highest value and the shortest main trajectory. The order of consideration of Zones and particular trajectories is determined by the grammar of translations.

Next move, 1. ... a6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8 b7:c8 (Fig. 19, left). Symbol “:” means the removal of element. Here the grammar cuts this branch with the value of -1 (as a win of the Black side).

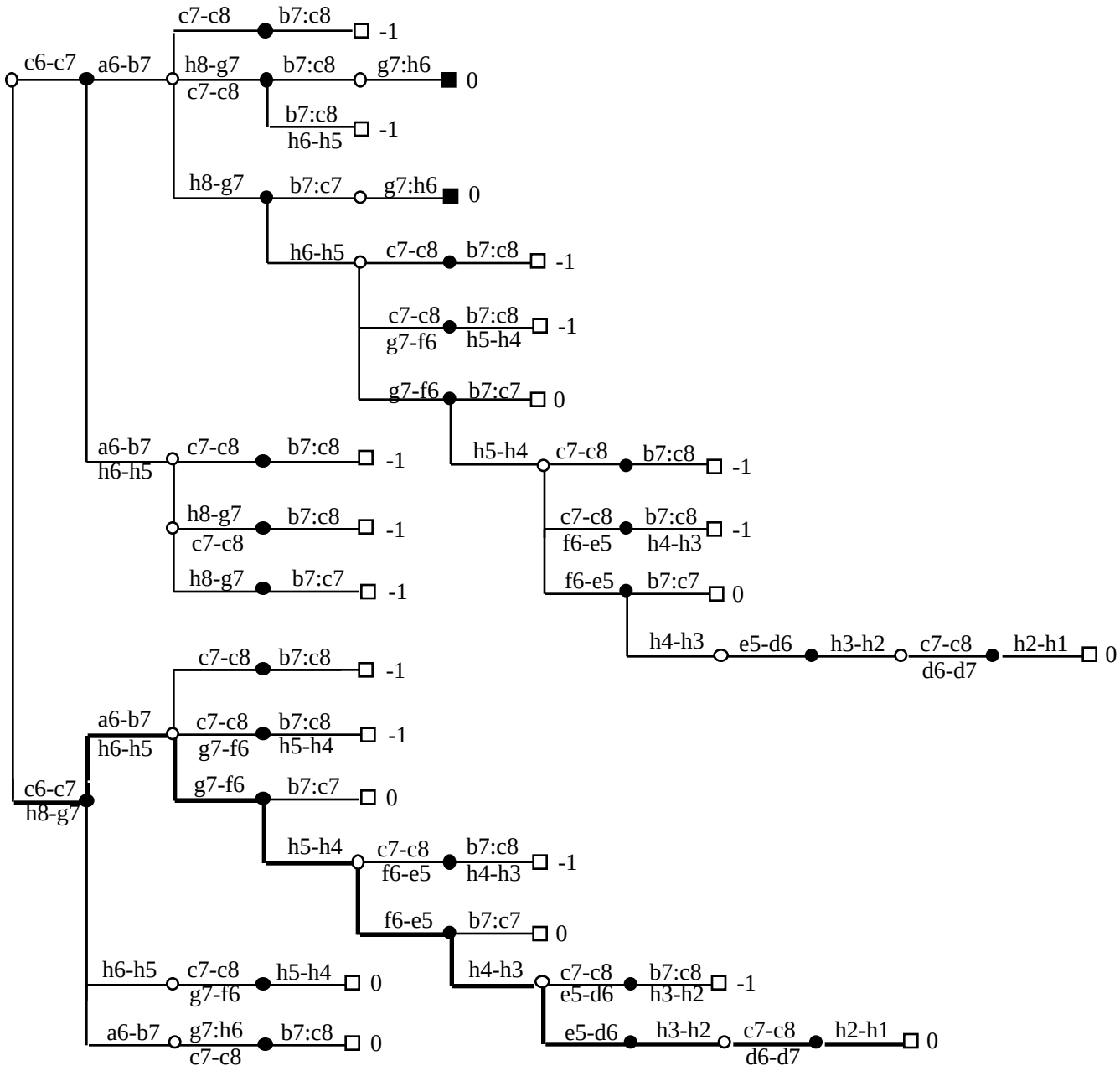

Fig. 17. Search tree for the new 2D optimization problem with concurrent motions.

This value is given by the special procedure of "generalized square rules" built into the grammar. This procedure determined that W-FIGHTER is out of the Zone of B-BOMBER, thus it can not intercept B-BOMBER.

Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search. After climb up to the move 1. ... a6-b7, the tree to be analyzed consists of one branch (of two plies): 2. c7-c8 b7:c8. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange on c8 (in favor of the White side). This can be achieved by participation of W-FIGHTER from h8, i.e., by generation and inclusion of the new so-called "control" Zones with the main trajectory from h8 to c8. These Zones have been detected (within the horizon 5) in the terminal state after the move 2. ... b7:c8 Fig. 19 (left).

Obviously they could not be detected in the initial state of this problem (Fig. 18) because the main element, W-BOMBER, could not “see” the target, B-FIGHTER, within given horizon.

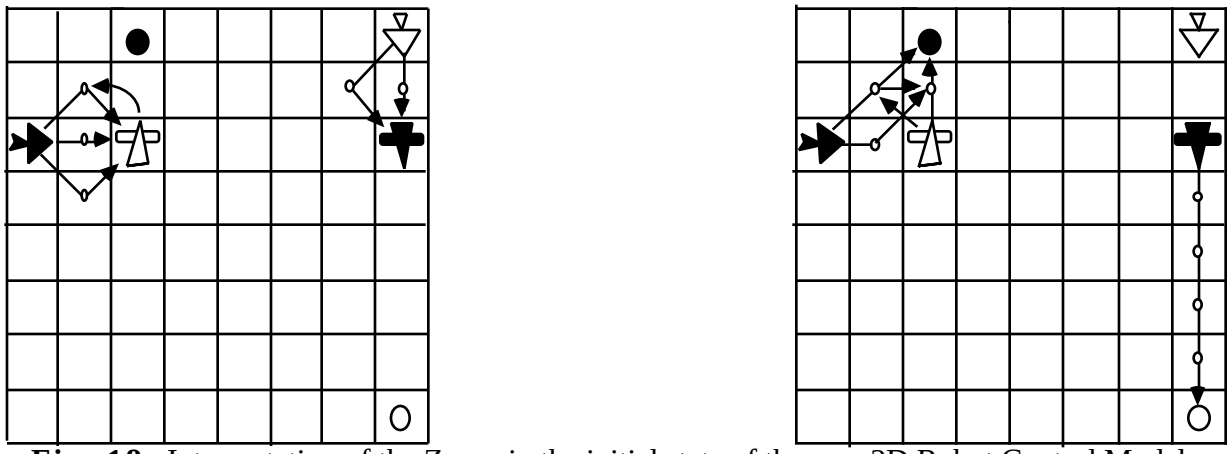

Fig. 18. Interpretation of the Zones in the initial state of the new 2D Robot Control Model.

However, at the moment of detection it was too late to include them into the search. These Zones have been stored for possible activation at the higher levels of the search tree. The set of different Zones from h8 to c8 (the bundle of Zones) is shown in Fig. 19 (right).

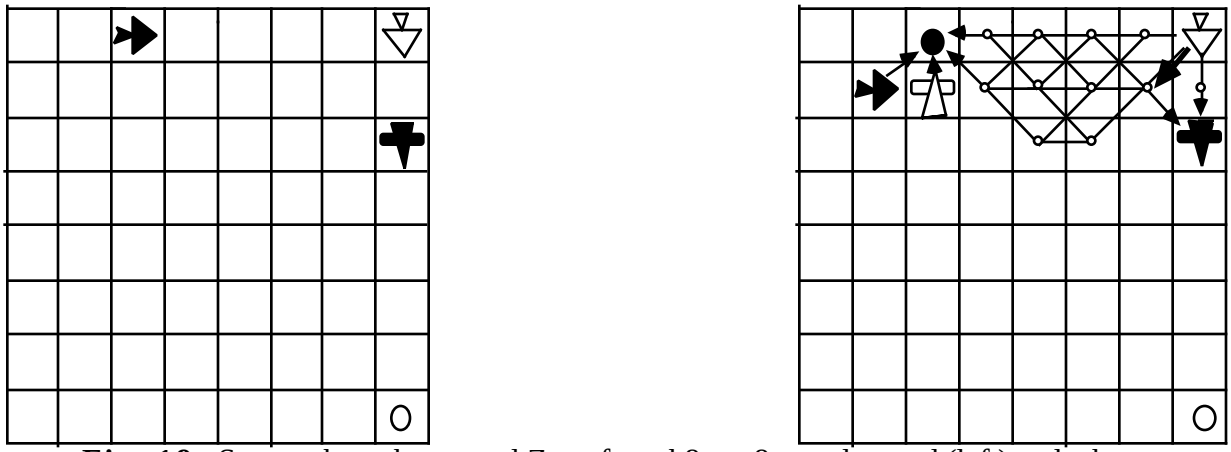

Fig. 19. States where the control Zone from h8 to c8 was detected (left) and where it was included into the search (right)

The move-ordering procedure picks the subset of Zones with main trajectories passing g7. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing W-BOMBER on h6. The motion along such trajectories allows to “gain time”, i.e., to approach two goals simultaneously.

The generation continues with the simultaneous motion of two agents, the double move, W-FIGHTER and W-BOMBER in their respective Zones: 2. h8-g7/c7-c8. The B-FIGHTER intercepts W-BOMBER at c8: 2. ... b7:c8. Now W-FIGHTER is ready to destroy B-BOMBER moving along the attacking trajectory ***a***(7)***a***(h6): 3. g7:h6. In this state all the BOMBERs have been destroyed and the grammar evaluates it as a draw (0) and initiates the backtracking climb. Move 2. ... b7:c8 is changed for the double move 2. ... b7:c8/h6-h5. That way the grammar included motion in the Zone of B-BOMBER attacking unmovable target on h1. The “square rule” procedure terminated branch, evaluated it as a win (-1) for the Black side, and initiated backtracking climb. Analogously to the previous case, the inspection procedure determined that the current minimax value (-1) can be improved by the improvement of the exchange on c8. Again, this can be achieved by the inclusion of Zone from h8 to c7. Of course, the best “time-gaining” move in this Zone is 2. h8-g7, and now the grammar includes this move as a single one 2. h8-g7 postponing motion in the

Zone of W-BOMBER. In this state (Fig. 20, left) a set of new control Zones of W-FIGHTER from g7 to c7 have been detected and stored as idle to be activated later if necessary. In response, Black hit W-BOMBER at c7: 2. ... b7:c7 along the intercepting trajectory ***a***(b7)***a***(c7). W-FIGHTER continues its attack of B-BOMBER: 3. g7:h6. This state is evaluated as a draw.

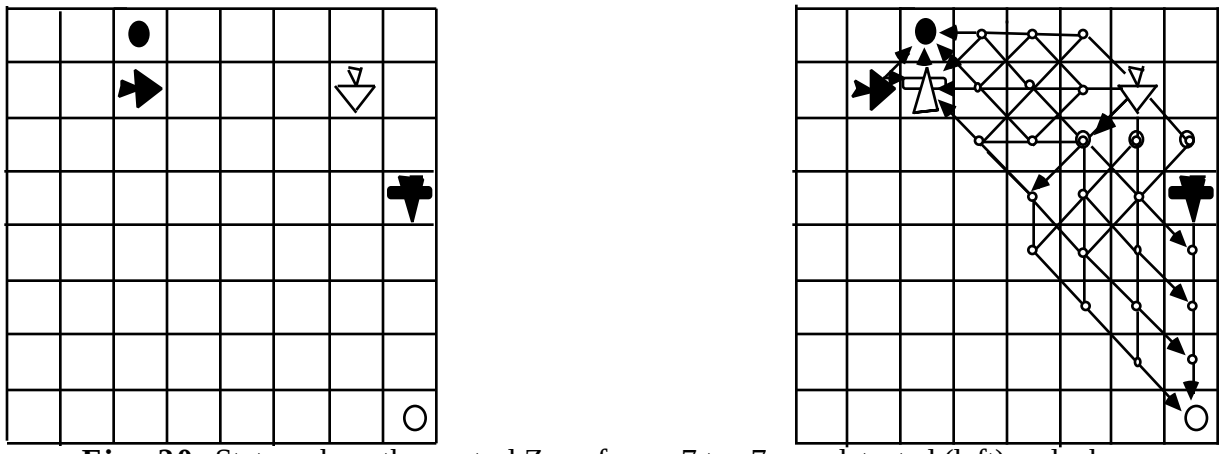

Fig. 20. States where the control Zone from g7 to c7 was detected (left) and where it was included into the search (right).

After the cut and climb, the inspection procedure included motion in the Zone of B-BOMBER instead of 2. ... b7:c7. New move 2. ... h6-h5 can not be included as a double move simultaneously with the old one because B-BOMBER can not move at the moment when W-BOMBER is being destroyed not at its final destination (c8). The following motion in 3. c7-c8 b7:c8 resulted in the termination of this branch with the value -1 in favor of Black.

New climb up to the move 2. ... h6-h5 and execution of the inspection procedure result in the inclusion of the groups of new control Zones from g7 to c7 and to c8 in order to improve the exchanges at these locations. Both groups of Zones (to c7 and c8) have been detected earlier in the search tree. The set of Zones with different main trajectories from g7 to c7 and from g7 to c8 is shown in Fig. 20 (right). Besides that, the trajectories from g7 to h4, h3, h2, and h1 are shown in the same Fig. 20. These are “potential” first negation trajectories. It means that beginning with the second symbol ***a***(f6), ***a***(g6) or ***a***(h6) these trajectories become first negation trajectories in the Zone of B-BOMBER on h5. Speaking informally, from squares f6, g6, and h6, Zone gateways, W-FIGHTER can intercept B-BOMBER (in case of White turn). The move-ordering procedure picks the subset of Zones with the main trajectories passing f6. These trajectories partly coincide with the potential first negation trajectories. The motion along such trajectories allows to “gain time”, i.e., to approach two goals simultaneously. Thus, the double move 3. c7-c8/g7-f6 is included.

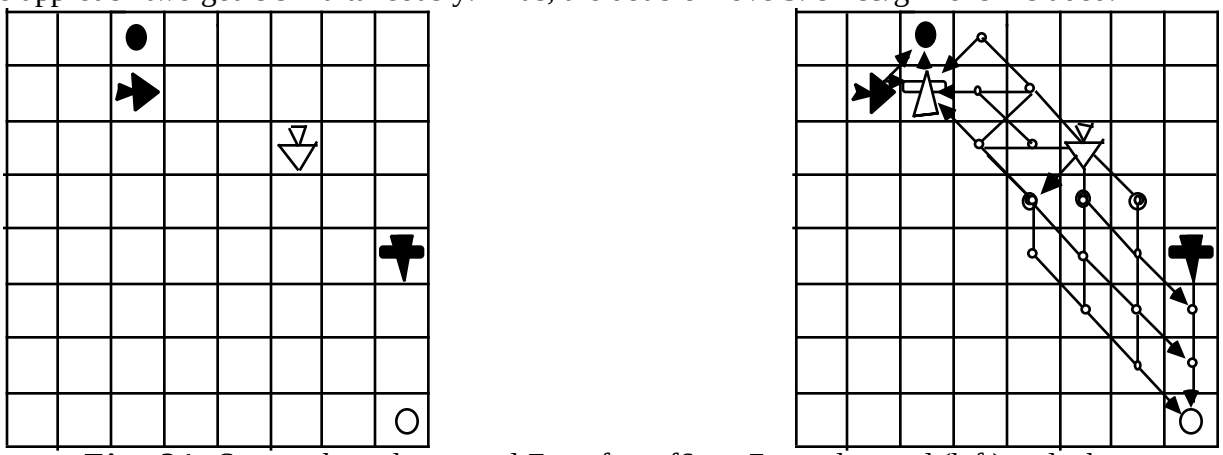

Fig. 21. States where the control Zone from f6 to c7 was detected (left) and where it was included into the search (right).

After the response 3. ... b7:c8/h5-h4, the branch was terminated with the value -1. The following climb and branching with inclusion of 3. g7-g6 as a single move resulted in 3. ... b7:c7 with terminal value 0. This state is shown in Fig. 21, left. The draw value has been assigned by the "square rule" procedure which detected that W-FIGHTER is in the Zone of B-BOMBER and thus has enough time for interception.

After the climb, Black side continued branching 3. ... h5-h4. The following tree generation until 4. f6-e5 is analogous with the previous one after 2. ... h6-h5. Move 4. f6-e5 is selected by the move ordering procedure as the time-gaining move approaching two goals simultaneously, c7 as a goal of the control Zone of W-FIGHTER and one of the gateways (e5, f5, g5) of the Zone of B-BOMBER (Fig. 21, right). After 6. c7-c8/d6-d7 h2-h1, it is terminated with the value of 0. The following climb with activation of the inspection procedure in every black node ended at 1. ... a6-b7 which has been changed for the double move 1. ... a6-b7/h6-h5. It seems that this move almost depreciated previous search. The minimax value brought to the top of the tree generated so far is -1. However, the tree generation followed after the change of 1. c6-c7 for the double move 1. c6-c7/h8-g7 showed that previous search was very important. As a result of this search the grammar *learned key networks*, Zones of W-FIGHTER with main trajectories from g8 to c8, from g7 to c7 and c8, and from f6 to c7.

After the change of 4. ... b7:c7 for 4. ... h4-h3, the following branch is pretty straightforward. These networks have been used successfully in a different context after 1. c6-c7/h8-g7. The optimal branch is shown in Fig. 17 with bold lines.

The total number of moves included into this tree is 63. The maximum depth reached is 12. This means that the branching factor [20] of this tree is 1.24, i.e., the search is highly goal-oriented. The unreduced branching factor for this problem is about 18, taking into account the allowance of simultaneous moves and no-motion move.

17 Totally Concurrent Robotic Model: Problem Statement

Robots with different moving capabilities are shown in Fig. 22. The operational district X is the table 8 x 8. Robot W-FIGHTER (White Fighter) standing on h8, can move to any next square (shown by arrows). The other robot B-BOMBER (Black Bomber) from h7 can move only straight ahead, one square at a time, e.g., from h7 to h6, from h6 to h5, etc. Robot B-FIGHTER (Black Fighter) standing on a6, can move to any next square similarly to robot W-FIGHTER (shown by arrows). Robot W-BOMBER (White Bomber) standing on c6 is analogous with the robot B-BOMBER; it can move only straight ahead but in reverse direction. Thus, robot W-FIGHTER on h8 can reach any of the points y {h7, g7, g8} in one step, i.e., $R_{W\text{-}FIGHTER}(h8, y)$ holds, while W-BOMBER can reach only c7 in one step.

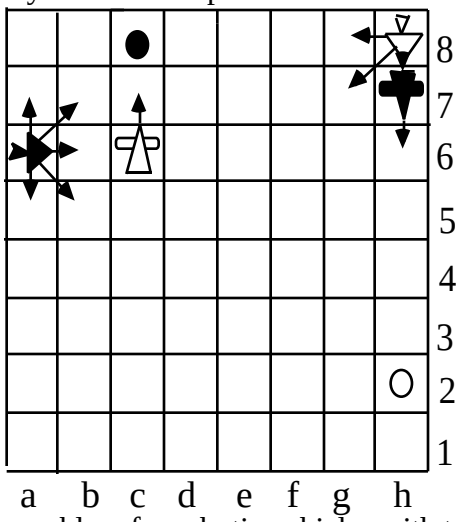

Fig. 22. 2D optimization problem for robotic vehicles with totally concurrent motions.

Assume that robots W-FIGHTER and W-BOMBER belong to one side, while B-FIGHTER and B-BOMBER belong to the opposing side: W-FIGHTER P_1, W-BOMBER P_1, B-FIGHTER P_2, B-BOMBER P_2. Also assume that two more robots, W-TARGET and B-TARGET, (unmoving devices or targeted areas) stand on h2 and c8, respectively. W-TARGET belongs to P_1, while B-TARGET P_2. Each of the BOMBERs can destroy unmoving TARGET ahead of the course. Each of the FIGHTERs is able to destroy an opposing BOMBER approaching its location, but it also able to destroy an opposing BOMBER if this BOMBER itself arrives at the current FIGHTER's location. For example, if the B-FIGHTER is at location c8 and W-BOMBER arrives there (unprotected) then during the same time increment it destroys the TARGET and is destroyed itself by B-FIGHTER. Each BOMBER can be protected by its friendly FIGHTER by approaching BOMBER's prospective location. In the latter case the joint protective power of the combined weapons of the friendly BOMBER and FIGHTER can protect the BOMBER from interception. For example, W-FIGHTER located at d6 can protect W-BOMBER on c6 and c7.

Each of the BOMBERs is vulnerable not only to a FIGHTER's attack but also to the explosion of another BOMBER. If W-FIGHTER hits B-BOMBER while the latter is fully armed, i.e., it is not at its final destination – square h2, and W-BOMBER is moving during the same time increment, it will be destroyed as a result of the B-BOMBER's explosion. If it is not moving at this moment it is safe. Similar condition holds for B-BOMBER: it can not move at the moment when W-BOMBER is being destroyed (excluding c8).

The combat considered can be broken into two local operations. The first operation is as follows: robot B-BOMBER should reach point h2 to destroy the W-TARGET, while W-FIGHTER will try to intercept this motion. The second operation is similar: robot W-BOMBER should reach point c8 to destroy the B-TARGET, while B-FIGHTER will try to intercept this motion. After destroying the opposing TARGET and keeping the BOMBER safe, the attacking side is considered as a winner of the local operation and the global combat. The only chance for the opposing side to avenge is to hit its TARGET at the same time increment and this way end the battle in a ***draw***. The conditions considered above give us S_t, the description of target states of the Complex System. The description of the initial state S_i is obvious and follows from Fig. 22.

Assume that all the agents of the opposing sides can move ***simultaneously***. There is no alternation of turns. It means, for example, that during the current time increment, all the four vehicles, W-BOMBER, W-FIGHTER, B-BOMBER, and B-FIGHTER, three of them, two, one, or none of them can move. This means that this is a model with ***incomplete information*** about the current move (before it is done). When moving each side does not know the opposing side component of the concurrent move, i.e., the immediate opposing side motions, if they are not constrained to one or zero motions and, thus, can be predicted. Moreover, after developing a strategy each side can not follow it because of the uncertainty with the other side current motions. However, it the strategy includes only variations of concurrent moves with single "universal" component (group of motions) for one side good for all possible components of the other side, this strategy can be actually implemented.

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-BOMBER from h7 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. ***Is there a strategy for the White side to make a draw*****?**

18 Totally Concurrent Robotic Model: Search Generation

Consider how the hierarchy of languages works for the optimal control of this model. We have to generate the Language of Trajectories and the Language of Zones in each state of the search. We generate a string of the Language of Translations [50] representing it as a search tree (Fig. 23) and comment on its generation. This tree is different from conventional search trees. Every concurrent move is represented by *two consecutive arcs*. The arc outgoing the white node represents the

White component of a concurrent move, the concurrent motions of the White side, while the arc outgoing the black node represents the Black component of the same move.

Fig. 23. Search tree for the Totally Concurrent Model.

First, the Language of Zones in the start state is generated. Every aircraft tries to attack every opposing side aircraft. The targets for attack are determined within the limit of five steps. It means that horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of main trajectories of all Zones must not exceed 5 steps. All the Zones generated in the start state are shown in Fig. 24. Zones for FIGHTERs as attacking elements are shown in the left diagram, while Zones for BOMBERs – in the right one.

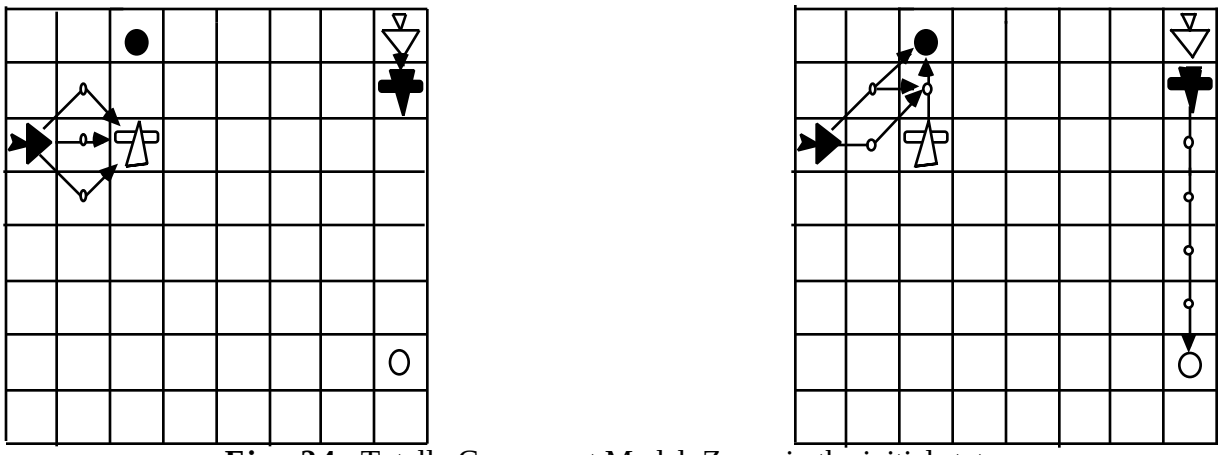

Fig. 24. Totally Concurrent Model: Zones in the initial state

Generation begins with the concurrent move 1. c6-c7 a6-b7 in the White Zone with the vulnerable Black target of the highest value and the shortest main trajectory. The order of consideration of Zones and particular trajectories is determined by the grammar of translations.

The Black component of this move, 1. ... a6-b7, is in the same Zone along the first negation trajectory. The interception continues: 2. c7-c8 b7-c8/h7-h6 (Fig. 25, left). This is a triple move. During the second time increment W-BOMBER hit the TARGET at c8 and was destroyed by the B-FIGHTER at c8. Also, immediately, the attack Zone of the B-BOMBER from h7 to h2 was activated: h7-h6 is the motion during the same time increment. Here the grammar terminates this branch with the value -1 (as a win of the Black side). This value is given by the special branch termination procedure built into the grammar. This procedure determined that W-FIGHTER is out of the Zone of B-BOMBER, thus, it can not intercept B-BOMBER which means that the latter will successfully hit the TARGET on h2.

Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated so far. After climb up to the move 1. c6-c7 a6-b7, the subtree to be analyzed consists of one branch (of one move): 2. c7-c8 b7-c8/h7-h6. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange on c8 (in favor of the White side). This can be achieved by participation of W-FIGHTER from h8, i.e., by generation and inclusion of the new so-called "control" Zones with the main trajectory from h8 to c8. These Zones were detected (within the horizon 5) in the terminal state after the move 2. c7-c8 b7-c8/h7-h6, Fig. 25 (left).

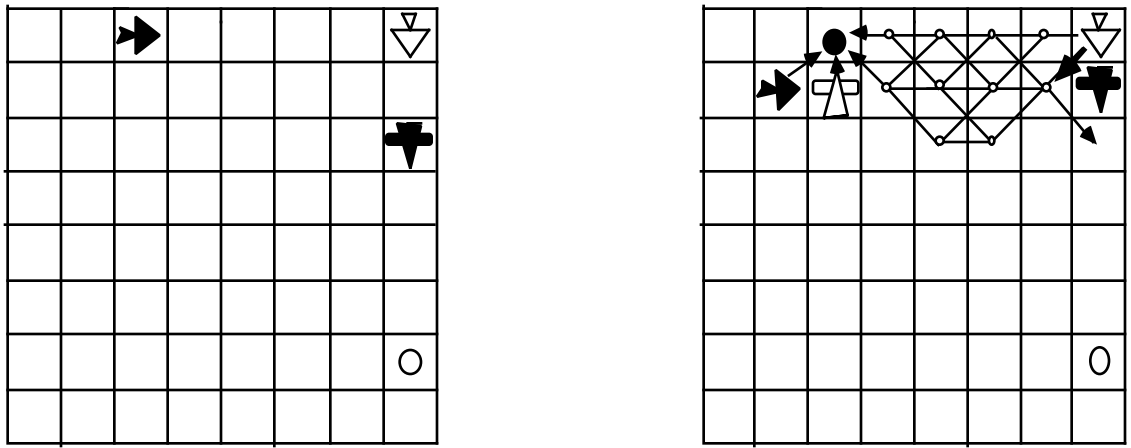

Fig. 25. States where the control Zone from h8 to c8 was detected (left) and where it was included into the search (right)

Obviously they could not be detected in the initial state of this problem (Fig. 24) because the main element, W-BOMBER, could not “see” the target, B-FIGHTER, within given horizon. Also, at the moment of detection it was too late to include them into the search. These Zones have been stored and kept idle for possible activation at the higher levels of the search tree. The set of different Zones from h8 to c8 (the bundle of Zones) is shown in Fig. 25 (right). The move-ordering procedure picks the subset of Zones with main trajectories passing g7. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing W-BOMBER on its future location h6. The motion along such trajectories allows to “gain time”, i.e., to approach two goals simultaneously.

The generation continues with the simultaneous motion of all four agents, the four-move, W-BOMBER, W-FIGHTER and B-FIGHTER, B-BOMBER, in their respective Zones: 2. c7-c8/h8-g7 b7-c8/h7-h6. The B-FIGHTER intercepted W-BOMBER at c8 while W-FIGHTER is unable to intercept the B-BOMBER during its attack from h6 to h2. The branch termination procedure determined that W-FIGHTER is outside the B-BOMBER’s attack Zone, terminated this branch, evaluated it as a win for the Black (-1), and initiated the backtracking climb. Move 2. ... was changed for the triple move 2. h8-g7 b7-c8/h7-h6 in attempt to find a better combination of White motions.

Black side, after finding b7-c8/h7-h6 to be a “good” component of the concurrent move 2. in the previous branches, continues to include this component in the following branches. Obviously, this component is very important. As it was noted above, a totally concurrent model is a model with incomplete information. Each side knows all the previous moves, the history of operation, and, theoretically, all possible future outcomes of the current move, the look-ahead tree. The only thing it does not know is the concurrent action of the opposing side as a component of the current move. Thus, for each side it is important to find not just a “good” own component of a concurrent move but a component to be “good” for all components of the opposing side. Such component would allow to avoid uncertainty in constructing an optimal variation, a branch, which can be implemented. A component b7-c8/h7-h6 is a candidate to be a good one for the Black while h8-g7 is a candidate for White.

After 2. h8-g7 b7-c8/h7-h6 termination procedure did not terminate the branch, and continued 3. c7-c8 h6-h5 in the same Black and White Zones. Then it terminated the branch and evaluated it as a win (-1) for the Black side (Fig. 26, left). Indeed, W-BOMBER hit B-TARGET on c8 but it is being destroyed itself by B-FIGHTER which was waiting for it at c8. Also, W-FIGHTER again is out of the attack Zone of B-BOMBER from h5 to h2. In this state a set of new control Zones of W-FIGHTER from g7 to c8 were detected and stored as idle to be activated later if necessary.

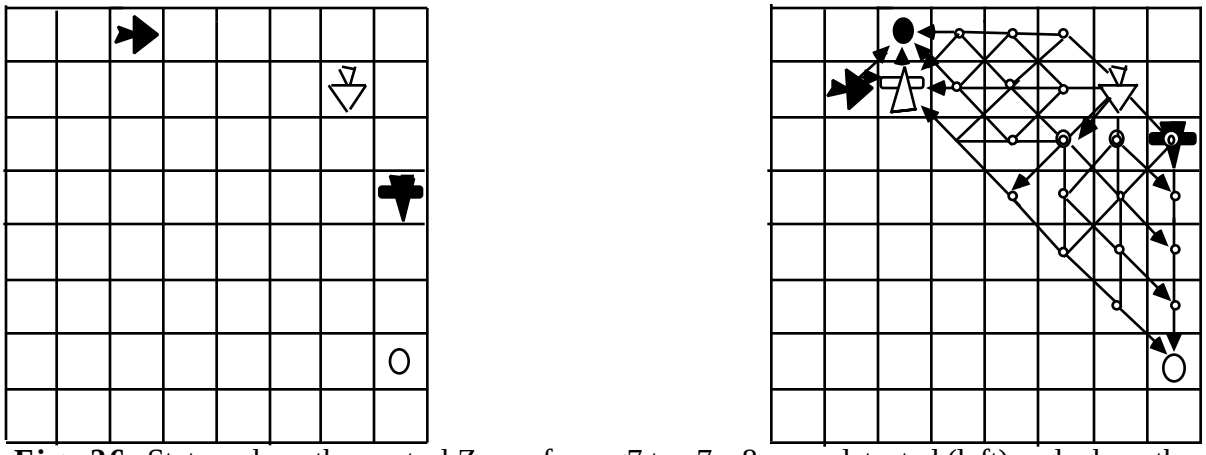

Fig. 26. States where the control Zones from g7 to c7, c8 were detected (left) and where they were included into the search (right)

New climb up to the move 2. h8-g7 b7-c8/h7-h6 and execution of the inspection procedure result in the inclusion of the groups of new control Zones from g7 to c7 and c8 in order to improve

the exchanges at these locations. Both groups of Zones (to c7 and c8) have been detected earlier in the search tree. The set of Zones with different main trajectories from g7 to c7 and from g7 to c8 is shown in Fig. 26 (right). Besides that, the trajectories from g7 to h4, h3, and h2, are shown in the same Fig. 26. These are “potential” first negation trajectories. It means that beginning with the second symbol ***a***(f6), ***a***(g6) or ***a***(h6) these trajectories become first negation trajectories in the Zone of B-BOMBER on h6. Speaking informally, from the squares f6, g6, and h6, Zone gateways, W-FIGHTER can intercept B-BOMBER. The move-ordering procedure picks the subset of Zones with the main trajectories passing f6. These trajectories partly coincide with the potential first negation trajectories. The motion along such trajectories allows to “gain time”, i.e., to approach two goals simultaneously.

Thus, the new White component 3. c7-c8/g7-f6 is included with the same Black component 3. ... h6-h5, the branch was terminated with the value -1. The following climb and branching with inclusion of g7-f6 as a single motion component resulted in 3. g7-f6 h6-h5, and the branch is not terminated. It continues with the move 4. c7-c8 h5-h4. This state is shown in Fig. 27, left. Then this branch is terminated with the value -1. As usual, this value was assigned by the termination procedure which detected that W-FIGHTER is outside the Zone of B-BOMBER and thus does not have enough time for interception.

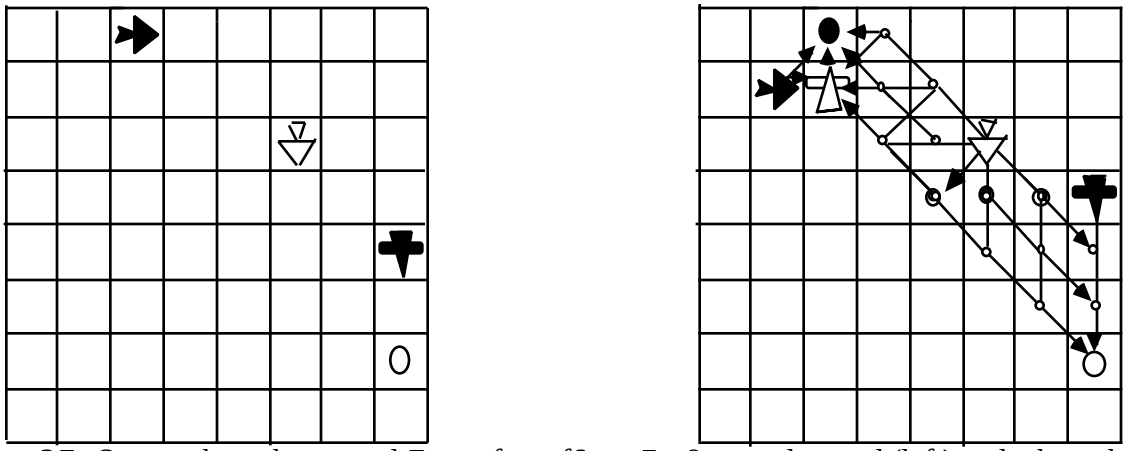

Fig. 27. States where the control Zones from f6 to c7, c8 were detected (left) and where they were included into the search (right).

After the climb, the grammar continued branching 4. c7-c8/f6-e5 h5-h4. The component f6-e5 is selected by the move ordering procedure as the time-gaining move approaching two goals simultaneously, c7 as a goal of the control Zone of W-FIGHTER and one of the gateways (e5, f5, g5) of the Zone of B-BOMBER (Fig. 27, right). But it was also terminated with the value -1. After 4. f5-e5 h4-h3, 5. e5-d6 h4-h3, and 6. c7-c8/d6-d7 h3-h2, the branch is terminated with the value of 0.

It seems that the sought draw is found. The following climb with activation of the inspection procedure in every node ended at the top level. All the attempts of the Black to change the components 4. ... h5-h4, 3. ... h6-h5, 2. ... h7-h6 for a different motion failed. If B-BOMBER's motion is not included in these concurrent moves the W-FIGHTER appears in the B-BOMBER's attack Zone and these branches should be terminated with the value 0 which does not improve the current minimax value for Black.

The Black component of 1. c6-c7 a6-b7 was changed for the double motions 1. c6-c7 a6-b7/h7-h6. It seems that this move almost depreciated previous search. The minimax value brought to the top of the subtree outgoing this move is -1. However, the tree generation followed after the change of 1. c6-c7 a6-b7/h7-h6 for the double move 1. c6-c7/h8-g7 a6-b7/h7-h6 showed that previous search was very important. As a result of this search the grammar *learned key networks*, Zones of W-FIGHTER with main trajectories from g8 to c8, from g7, f6 to c7 and c8. The optimal branch is shown in Fig. 23 with bold lines.

19 Discussion

Examples considered in Sections 9-18 demonstrate power of the Linguistic Geometry tools that allowed to transfer heuristics discovered in one problem domain, specifically, in the game of chess, to another domain of simplified aerospace robotic vehicles. It is even more interesting that search reduction achieved in the original domain with one-at-a-time motion of every movable unit multiplied tremendously in the new domain with the allowance of *concurrent* moves. We considered application of Linguistic Geometry to the problems of control of remotely piloted aircraft in conjunction with generation of the optimal combat scenario. These problems are computationally hard for conventional approaches.

In the first simplified example (Sections 10-12) the aircraft move in a serial mode, one aircraft at a time, and motions of opposing sides alternate. The number of legal moves in each state is equal to 9. Indeed, a FIGHTER has 8 legal moves if it is located inside X, not at the edges. Also, every BOMBER has one legal motion straight ahead. The depth of the search tree required to solve this problem should be at least 7. Thus, the total number of moves to be included into the search should be at least $9^7 \approx$ 5 million or, employing the minimax algorithm with alpha-beta cut-offs, it should not be less than $(9^7)^{1/2} \approx 2{,}200$ [20]. (This theoretical lower bound for the alpha-beta algorithm is practically unreachable.) The tree generated employing Linguistic Geometry tools contains 49 moves which means that the branching factor is reduced from 9 to 1.5!

After relaxation of the strictly serial mode (Sections 13-16) all the aircraft of each side could move simultaneously if necessary. Because of the allowance of concurrent motions the number of legal moves in each state grows significantly. This number is 18 taking into account that all the combinations of moves of one side and no-motion option. This number causes a tremendous growth of the state space. As usual, the total size of the state space grows exponentially (with the average number of legal moves in each state as a base). The total number of moves to be included in the search tree in order to solve new problem from Section 15 ($18^{12} \approx 5 \times 10^{14}$) is by far greater than the number ($9^7 \approx$ 5 million) required for the original serial problem. The actual number of moves (63) generated by the grammar of searches for the problem with partial concurrency is almost the same as in the serial case (49), and the branching factor even decreased from 1.5 to 1.24. The search reduction achieved in this example is even more dramatic than it was in the serial case.

In the last example (Sections 17-18) all the aircraft, cooperating and opposing, can move concurrently. This results in an extremely steep growth of the branching factor because all the combinations of simultaneous motions are legal. Indeed, the average number of legal motions for each side, i.e., the average number of different legal components of every concurrent move, is 18. Thus, the average number of legal moves in each state, the unreduced branching factor, is $18 \times 18 = 324$ (!), taking into account all the combinations of legal components. Another difficulty of this example is that each side when moving is uncertain about the concurrent motions of the other side. The Linguistic Geometry tools solved this problem demonstrating a dramatic reduction of the branching factor, to the value close to one. The total number of moves included in this search tree is 34. The maximum depth reached is 6. This means that the branching factor of this tree is 1.53, i.e., the search is highly goal-oriented. Obviously, 34 is a dramatic reduction in comparison with a 324^6 move tree that would have to be generated by conventional search procedures, or even with the theoretical minimum of the minimax search with alpha-beta cut-offs $(324^6)^{1/2} = 18^6 \approx 34$ million.

Looking at the complexity of the hierarchy of languages which represents each state in the search process, it is easy to speculate that the growth from the serial case to the concurrent one is represented by a linear function limited by multiplication to a constant factor close to one. This means that the entire algorithm did not lose its efficiency after allowing the concurrent moves, and this allowance is, probably, inherent to the Complex Systems and the Hierarchy of Formal Languages.

A series of simplified air combat problems considered here is still very close to the original

board game domain. It is possible to predict that the power of Linguistic Geometry goes far beyond these limits. The definition of the Complex System (see Section 5) is generic enough to cover a variety of different problem domains. The core component of this definition is the triple X, P, and R_p. Thus, looking at the new problem domain we have to define X, the finite set of points – locations of elements. We do not impose any constraints on this set while the aerospace operational district X considered in this paper as well as the original game board have different extra features, e.g., 2D space connectivity, which is totally unimportant for these problems. Thus, we can consider X, for example, as a set of orbits where the elements are in constant motion with respect to each other. The set of elements P, e.g., movable units, in our problem is quite small, while their moving capabilities, binary relations of R_p, are non-sophisticated. Indeed, during one time interval our aircraft can move only to the next area. An example of the problem with greater number of movable units with more advanced moving capabilities is considered in [58]. The definition of R_p is exactly the place for introducing the variable speed, the gravity impact, the engine impulse duration, etc.

A dramatic search reduction achieved in the serial and concurrent cases allowed us to initiate the development of a prototype of the system for simulation and control of the real world aerospace combat with participation of aircraft, satellites, and unmanned aerial vehicles (UAVs). This work is currently under way at Phillips Lab, Kirtland AFB, NM, USA. The development of Linguistic Geometry towards aerospace multiagent systems will encompass the discovery of geometrical properties of subsystems, details of interactions between the agents within subsystems and between different subsystems, the effect of this complex hierarchical structure on the search reduction.

References

1. Rodin E. Semantic Control Theory, *Applied Mathematical Letters*, (pp. 73-78), 1(1), 1988.
2. Lirov Y., Rodin, E.Y., McElhaney, B.G., and Wilbur, L.W. Artificial Intelligence Modeling of Control Systems, *Simulation*, 1988, 50(1): 12-24.
3. Garcia-Ortiz, A. et al. Application of Semantic Control to a Class of Pursue-Evader Problems, *Computers and Mathematics with Applications*, 1993, 26(5): 97-124.
4. Shinar, J. Analysis of Dynamic Conflicts by Techniques of Artificial Intelligence, INRIA Report, Antipolis, 1990.
5. Garey, M.R. and D.S.Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, 1991.
6. Strosnider, J.K. and Paul, C.J. (1994). A Structured View of Real-Time Problem Solving, *AI Magazine*, 1994, 15(2): 45-66.
7. Leitmann, G. *Optimization Techniques with Applications to Aerospace Systems*, Academic Press, 1990.
8. Drabble, B. Spacecraft Command and Control Using Artificial Intelligence Techniques, *J. of the British Interplanetary Society*, 1991, 44: 251-254.
9. Pigeon, A., Howard, G., and Seaton B. Operational Aspects of Spacecraft Autonomy, *J. of the British Interplanetary Society*, 1992, 45: 87-92.
10. Chung, J., Liu, J. and Lin, K. Scheduling Periodic Jobs That Allow Imprecise Results. *IEEE Transactions on Computers*, 1990, 39(9): 1156-1174.
11. Korf, R.E. Real-Time Heuristic Search, *Artificial Intelligence*, 1990, 42(2-3): 189-211.
12. Lesser, V.R., Pavlin, J., and Durfee, E. Approximate Processing in Real-Time Problem Solving, *AI Magazine*, 1988, 9(1): 49-62.
13. Boddy, M. and Dean, T. Solving Time-Dependent Planning Problems, *Proc. of the 11th Int. Joint Conf. on AI*, 1989, 979-984.
14. Albus, J. Outline for a Theory of Intelligence. *IEEE Trans. on Systems, Man and Cybernetics*, 1991, 3: 473-509.
15. Knoblock, C.A. Learning Abstraction Hierarchies for Problem Solving, *Proc. of the 8th AAAI Conf.*, Menlo Park, CA, 1990, 923-928.

16. Mesarovich, M.D. and Takahara Y. *Abstract Systems Theory*, Berlin: Springer-Verlag, 1989.
17. Botvinnik, M.M. *Computers in Chess: Solving Inexact Search Problems.* Springer Series in Symbolic Computation, New York: Springer-Verlag, 1984.
18. McAllester, D. and Rosenblitt, D. Systematic Non-Linear Planning, *Proc. of AAAI-91*, 1991, 634-639.
19. Chapman, D. Planning for conjunctive goals. *Artificial Intelligence* 32(3), 1987.
20. Nilsson, N.J. *Principles of Artificial Intelligence*, Palo Alto, CA: Tioga Publ, 1980.
21. Stefik, M. Planning and meta-planning (MOLGEN: Part 2), *Artificial Intelligence*, 1981, 2: 141-169.
22. Sacerdoti, E.D. The Nonlinear Nature of Plans, *Proc. Int. Joint Conference on Artificial Intelligence*, 1975.
23. Chomsky, N. Formal Properties of Grammars. in *Handbook of Mathematical Psychology*, eds. R.Luce, R.Bush, E. Galanter., vol. 2. New York: John Wiley & Sons, 1963, 323-418.
24. Ginsburg, S. *The Mathematical Theory of Context-Free Languages*, McGraw Hill, New York, 1966.
25. Fu, K.S. *Syntactic Pattern Recognition and Applications*, Prentice Hall, Englewood Cliffs, 1982.
26. Narasimhan, R.N. Syntax–Directed Interpretation of Classes of Pictures. *Comm. of ACM*, 1966, 9: 166–173.
27. Pavlidis, T. *Structural Pattern Recognition*, New York: Springer-Verlag, 1977.
28. Shaw, A.C. A Formal Picture Description Scheme as a Basis for Picture Processing System, *Information and Control*, 1969, 19: 9-52.
29. Feder, J. Plex languages. *Information Sciences*, 1971, 3: 225–241.
30. Rosenfeld, A. *Picture Languages, Formal Models for Picture Recognition*, Academic Press, 1979.
31. Knuth, D.E. Semantics of Context-Free Languages. *Mathematical Systems Theory*, 1968, 2: 127–146.
32. Rozenkrantz, D.J. Programmed Grammars and Classes of Formal Languages, *J. of the ACM*, 1969, 1: 107–131.
33. Volchenkov, N.G. The Interpreter of Context-Free Controlled Parameter Programmed Grammars. In L.T. Kuzin, Eds., *Cybernetics Problems. Intellectual Data Banks*, 1979, 147–157, USSR Academy of Sci., Moscow, (in Russian).
34. Fikes, R.E. and Nilsson, N.J. STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving. *Artificial Intelligence* 1971, 2: 189–208.
35. McCarthy, J. Circumscription–A Form of Non-Monotonic Reasoning. *ArtificialIntelligence* , 1980, 13: 27-39.
36. McCarthy, J. and Hayes, P.J. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 1969, 4: 463–502.
37. Stilman, B. The Computer Learns. in Levy, D., *1976 US Computer Chess Championship*, Computer Science Press, Woodland Hills, CA, 1977, 83-90.
38. Botvinnik, M., Petriyev, E., Reznitskiy, A., et al. Application of New Method for Solving Search Problems For Power Equipment Maintenance Scheduling. *Economics and Mathematical Methods*, 1983, 6: 1030-1041, (in Russian).
39. Reznitskiy, A.I. and Stilman, B. Use of Method PIONEER in Automating the Planning of Maintenance of Power-Generating Equipment. *Automatics and Remote Control*, 1983, 11: 147-153), (in Russian).
40. Stilman, B. Hierarchy of Formal Grammars for Solving Search Problems. In *Artificial Intelligence. Results and Prospects*, *Proceedings of the International Workshop*, Moscow, 1985, 63-72, (in Russian).
41. Stilman, B. A Linguistic Approach to Geometric Reasoning, *Int. J. Computers and Mathematics with Applications*, 1993, 26(7): 29-57.
42. Stilman, B. A Syntactic Structure for Complex Systems. *Proc. Second Golden West Int. Conf. on Intelligent Systems*, Reno, NE, June 1992, 269-274.
43. Stilman, B. A Geometry of Hierarchical Systems: Generating Techniques. *Proc. Ninth Israeli*

Conference on Artificial Intelligence and Computer Vision, Tel Aviv, Israel, Dec. 1992, 95-109.
44. Stilman, B. A Syntactic Approach to Geometric Reasoning about Complex Systems," *Proc. Fifth Int. Symp. on Artificial Intelligence*, Cancun, Mexico, Dec. 1992, 115-124.
45. Stilman, B. Network Languages for Complex Systems, *Int. J. Computers and Mathematics with Applications*, 1993, 26(8): 51-79.
46. Stilman, B. Syntactic Hierarchy for Robotic Systems, *Integrated Computer-Aided Engineering*, 1993, 1(1): 57-81.
47. Stilman, B. A Formal Language for Hierarchical Systems Control, *Languages of Design*, 1993, 1(4): 333-356.
48. Stilman, B. Hierarchical Network for Systems Control. *Proc. of the Tenth Israeli Symposium on Artificial Intelligence and Computer Vision*, Ramat Gan, Israel, Dec. 1993, 141-153.
49. Stilman, B. Knowledge Representation in Linguistic Geometry. *Proc. of the Fifth Int. UNB Artificial Intelligence Symposium*, Fredericton, New Brunswick, Canada, August 1993, 219-229.
50. Stilman, B. Translations of Network Languages. *Int. J. Computers and Mathematics with Applications*, 1994, 27(2): 65-98.
51. Stilman, B. A Formal Model for Heuristic Search. *Proc. of the 22nd Annual ACM Computer Science Conf.*, March 8-10, Phoenix, AZ, 1994, 380-389.
52. Stilman, B. A Linguistic Geometry for Intelligent Autonomous Systems. *Cybernetics and Systems-94. Proc. of the Twelfth European Meeting on Cybernetics and Systems Research*, Vienna, Austria, April 1994, 1483-1490.
53. Stilman, B. A Linguistic Geometry for Space Applications, *Proc. of the 1994 Goddard Conference on Space Applications of Artificial Intelligence*, NASA Goddard Space Flight Center, Greenbelt, MD, USA, May 1994, 87-101.
54. Stilman, B. Heuristic Networks for Space Exploration, *Telematics and Informatics, An Int. Journal on Telecommunications & Information Technology*, (invited paper), 1994, 11(4), 403-428.
55. Stilman, B. A Linguistic Geometry of the Chess Model, *Advances in Computer Chess 7*, 1994, 91-117.
56. Stilman, B. Network Languages for Intelligent Control, *An International Journal: Computers & Mathematics with Applications.*, 1995, (to appear).
57. Stilman, B., A Linguistic Geometry for Control Systems Design, *International Journal of Computers and Their Applications*, (invited paper), Vol. 1, No. 2, 89-110, Dec. 1994.
58. Stilman, B., Deep Search in Linguistic Geometry, Symposium on LINGUISTIC GEOMETRY AND SEMANTIC CONTROL, *Proc. of the First World Congress on Intelligent Manufacturing: Processes and Systems*, 868-879, Mayaguez, Puerto Rico, Feb. 1995.
59. Yakhnis, V., Stilman, B., Foundations of Linguistic Geometry: Complex Systems and Winning Conditions, Symposium on LINGUISTIC GEOMETRY AND SEMANTIC CONTROL, *Proc. of the* First World Congress on Intelligent Manufacturing: Processes and Systems, 843-854, Mayaguez, Puerto Rico, Feb. 1995.
60. Yakhnis, V., Stilman, B., A Multi-Agent Graph-Game Approach to Theoretical Foundations of Linguistic Geometry, *Proc. of the Second World Conference on the Fundamentals of Artificial Intelligence* (WOCFAI 95), Paris, France, July 1995.
61. Stilman, B., Multiagent Air Combat with Concurrent Motions, Symposium on LINGUISTIC GEOMETRY AND SEMANTIC CONTROL, Proc. of the First World Congress on Intelligent Manufacturing: Processes and Systems, 855-867, Mayaguez, Puerto Rico, Feb. 1995.