Proceedings of DETC'98
1998 ASME Design Engineering Technical Conferences
September 13-16, 1998, Atlanta, Georgia, USA

DETC98/DAC-5628

3D SPATIAL LAYOUTS USING A-TEAMS

Sanjay Sachdev
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: sachdev+@cmu.edu

Christiaan J. J. Paredis
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: Paredis+@cmu.edu

Satyandra K. Gupta
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: skgupta@ri.cmu.edu

Sarosh N. Talukdar
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: talukdar@ece.cmu.edu

ABSTRACT

Spatial layout is the problem of arranging a set of components in an enclosure such that a set of objectives and constraints is satisfied. The constraints may include non-interference of objects, accessibility requirements and connection cost limits. Spatial layout problems are found primarily in the domains of electrical engineering and mechanical engineering in the design of integrated circuits and mechanical or electromechanical artifacts. Traditional approaches include ad-hoc (or specialized) heuristics, Genetic Algorithms and Simulated Annealing. The A-Teams approach provides a way of synergistically combining these approaches in a modular agent based fashion. A-Teams are also open to the addition of new agents. Modifications in the task requirements translate to modifications in the agent mix. In this paper we describe how modular A-Team based optimization can be used to solve 3 dimensional spatial layout problems.

INTRODUCTION

A number of design and manufacturing problems require spatial arrangements of parts subject to a wide variety of spatial constraints. Given a set of arbitrarily shaped rigid 3D solid objects, an arbitrarily shaped housing (which may be optional in some cases), the goal is to arrange the objects inside the housing without violating any of the given spatial constraints while optimizing the given objective function(s). Examples include the design of satellites, the design of cars, packing for material handling, and 3D nesting for rapid prototyping. We refer to such problems as spatial layout problems. Different spatial layout problems have different types of constraints and objective functions. Most spatial layout problems have many different conflicting constraint and objective functions which make them very challenging, and the software currently used for solving such problems is difficult and expensive to develop.

There has been considerable research in 2D spatial layout problems found in integrated circuit design. Extensions of such problems to three dimensions consist of packing 2D layers. The objects are almost always rectangular in shape. Problems in material handling packing are characterized by packing box shaped containers. Most of the research in the Operations Research, Industrial Engineering and Electrical Engineering communities has been on packing such box-like objects, usually into box-like containers. Mechanical Engineering problems on the other hand are usually 3D in nature. However, there has not been much work in this area due to the lack of computational resources. It is only recently that computers have become powerful enough to attempt automated optimization of 3D spatial layouts. Recently developed technologies such as rapid prototyping (Hinzman 1995) also create a need for better 3D spatial layout tools.

Spatial layout problems in 3D are usually solved using ad-hoc heuristics, genetic algorithms (GA) or simulated annealing (SA). Ad-hoc heuristics take advantage of structure in the problem and are fast but fragile. GA and SA are more robust to differences in problem structure but much slower. Additionally, both methods optimize with respect to a single merit function, expressed as the weighted sum of all objectives and constraints, and users must decide on the relative importance of the various objectives a-priori.

We are developing a different approach to solving spatial layout problems that attempts to combine and extend the strengths of the existing methodologies. Our approach determines good spatial layouts of the physical components while satisfying requirements on component interference, accessibility, connectivity, and separation. The individual objectives and constraints are captured in software agents that interact with each other in an autonomous and asynchronous manner through shared memories containing candidate solutions. The agents are implemented in a distributed fashion and can be reconfigured dynamically, for instance to capture changes in the design requirements. To allow the designer to weigh the trade-offs in the constraints and optimization criteria, the Pareto optimal frontier of non-dominated solutions is computed and visualized. Our approach has the following benefits: (1) it has core technology components that can be used elsewhere; (2) additional constraints and objective functions can be easily added; (3) agents can be combined together in a variety of ways; (4) humans can participate in the problem solving process; and (5) the system is based on a distributed architecture which can use all the available computational power.

ASYNCHRONOUS TEAMS (A-TEAMS)

An A-Team (Talukdar and deSouza, 1994) is an organization of autonomous agents (or agents, for short). An agent is an entity whose input and output spaces are maintained by computers, and whose control system is completely self-contained (i.e. it decides for itself when and what to work on). Agents collaborate by modifying one another's results (trial solutions). The results collect in shared memories. An agent consists of five components: an input memory, a scheduler that determines when the agent will work, a selector tht chooses one or more solutions from the input memory, and an operator that modifies the selected solutions and writes the result to the output memory.

There are two classes of agents in A-Teams: constructors that produce new solutions by modifying old ones, and destroyers that erase old solutions. The organization of agents and memories can be represented as directed hypergraphs, where nodes represent memories and arcs represent agents. Such a graph is called a dataflow. A dataflow is

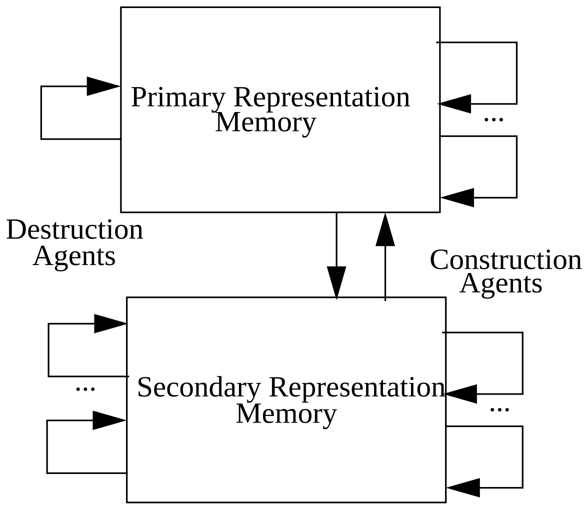
Primary Representation Memory
...
Destruction Agents
Construction Agents
...
Secondary Representation Memory
...

Figure 1. An A-Team consisting of two memories and several construction and destruction agents.

an A-Team if and only if all arcs lie in closed loops. A complete set of definitions, descriptions and prescriptions for A-Teams can be found in (Talukdar, 1998). Additional details and theoretical analyses of why A-Teams work well can be found in (Talukdar et al. 1998).

The power of A-Teams lies in their modular design. Solutions in various representations (or even solutions to related problems that might provide hints at solutions for the problem at hand) are stored in different memories. Heuristics that operate on these solutions are encapsulated as autonomous agents (they work independently and asynchronously of each other). There is no central controller as in hierarchical problem solvers. Thus agents can be added or removed as needed without affecting existing agents in the team. Each agent is independent of all others, and there may even be multiple copies of agents running simultaneously in parallel. Furthermore, the A-Team can be easily scaled as more processors become available, and the A-Team configuration may even be changed dynamically by adding and removing agents.

A-Teams have been developed for a variety of computation problems, including nonlinear equation solving (Talukdar et al. 1983), traveling salesman problems (deSouza, 1993), high rise building design(Quadrel, 1991), reconfigurable robot design (Murthy, 1992), diagnosis of faults in electric networks (Chen, 1992), control of electric networks (Talukdar and Ramesh, 1993), job shop scheduling (Chen et al., 1993), steel (Lee et al., 1995) and paper mill scheduling (Rachlin et al., 1996), train scheduling (Tsen, 1995), protein folding (Lukin et al. 1997) and constraint satisfaction (Gorti et al., 1996). Not only do these asynchronous teams find good solutions, but they appear to be scale effective: solution quality and speed improve with the addition of agents and computers.

SPATIAL LAYOUT - PROBLEMS AND METHODS

There are three primary areas of research in spatial layout problems. These are Operations Research (OR) and Industrial Engineering (IE) where the problems are usually referred to as "packing" problems; Electrical Engineering (EE), where the problems are called "layout" or "placement" problems; and Mechanical Engineering (ME), where the problems are related to "configuration design" problems. We will provide a brief survey of the work in all these areas, although our research focuses on spatial layout problems in Mechanical Engineering domains such as design, prototyping and manufacturing which typically require packing components with complicated shapes, subject to a variety of constraints.

Spatial Layout Problems

The focus in EE emphasizes connectivity costs between objects since there is some freedom on the size and shape of objects (which may lie within some neighborhood of the given target). The problems derive from the need to transform a circuit description, consisting of a set of modules and nets (wiring connecting the modules), into a layout in 2D which is then implemented in silicon to make integrated circuits. There are several steps to this process, consisting of: partitioning the circuit into components; placement of modules of fixed sizes without overlap onto a 2D surface representing the chip so as to minimize wiring requirements; routing the wiring around the modules on the surface; and compaction of the final arrangement. Placement usually implies locating rectangles of fixed size. Floorplanning is placement of rectangles that may vary in size. The main concern is to fit the rectangles and minimize the length (and quality, as measured for example by the number of bends) of wiring used to connect components. A good current survey for EE problems is found in (Aarts et al., 1997). Problems in EE are primarily 2D due to the planar nature of the integrated circuit technology. There is some extension into the third dimension, but this consists of stacking layers of components, which can be considered as a set of 2D problems. The objects and container are rectangular in shape. Such problems consider much simpler objects than the irregular, non-convex, 3D objects found in electromechanical assemblies.

The focus in OR, IE and ME has been mainly on maximizing packing density since the components in mechanical assembly and container packing are rigid. Other constraints such as stability or accessibility are usually ignored. Much of the work in OR is on 2D problems related to theoretically oriented bin-packing or knapsack problems. This work focuses on packing rectangles in rectangular boundaries. The OR literature focuses on the development of heuristics (for 2D or 3D box-like objects in box-like housings). Nonlinear programming methods are not used due to the extreme difficulty in problem formulation and the disconnected character of feasible regions in spatial layout problems. However there has been some exploratory investigation into other methods such as Simulated Annealing or Tabu Search. A general survey of packing problems in OR is (Dowsland and Dowsland, 1992). It describes the main types of problems that have been considered in the literature, classifying the problems by dimension (1, 2 or 3D). This paper also describes the use of non-heuristic based methods (such as branch and bound) for 2D rectangle packing, which require considerable computation time even for small problems. The literature focuses on packing density, usually ignoring other (possibly important) constraints such as accessibility or separation requirements. The 3D problems in OR or IE usually involve packing cuboids into other cuboids (e.g. for packing boxes in containers for shipping). In addition to the packing constraints, the additional constraint of locating the center of gravity of the packed container low and near the center (in the horizontal plane) of the container is often considered.

Spatial Layout Methods

There are three main categories of methods for packing rigid objects: Ad-Hoc (or specialized) Heuristics, Genetic Algorithms, and Simulated Annealing. Gradient based optimization methods such as nonlinear programming methods are normally not applied to layout problems because of the difficulty in setting up the required problem formulation.

Ad-Hoc Heuristics: These are most widely used in the OR and IE literature. For three dimensional problems, (Dowsland and Dowsland, 1992) states that the approaches are "entirely composed of a series of ad-hoc heuristic rules derived from common sense" but that "practical experience suggests that any of [these] methods will out-perform manual methods on average." The section on non-rectangular packing indicates the absence of methods here, with most of the work being in circle and sphere packing, primarily in the mathematical and recreational literature. It is pointed out that existing heuristics designed for rectangles are difficult to extend to these problems. Most heuristics locate the objects one at a time in the container, with some kind of layering rule to indicate how objects will be placed. The search space is usually limited to placing objects so that their corners align with already placed objects. Examples of such heuristics can be found in (Lengauer, 1990; Aziz, 1991; Gehring et al., 1990; Haessler and Talbot, 1990).

Genetic Algorithms (GA): These are modeled on the principle of natural selection (Davis, 1991). Solutions are represented as binary strings (chromosomes). Populations of candidate solutions are maintained. Members of the populations are ranked based on a "fitness function," which is usually the merit function being optimized. New solutions are generated by combining two or more solutions (crossover) and by perturbing solutions (mutation). This process, called reproduction, is modeled on natural selection. Solutions with better fitness have a better chance of being selected for reproduction. Genetic algorithms might either store the solution directly, or might make use of an encoded representation along with a decoder which maps from the solution string to an actual solution. For example, in packing problems, the solutions might consist of the location and orientation of the objects (direct representation), or might be an ordering of the objects (encoded representation) which is then mapped into an actual location. Alternatively, a combination of the two representations may be used such as an ordering and orientation for the objects, where the encoded part is used only to determine translations. Genetic Algorithm approaches to packing typically use partially or completely encoded representations with a packing heuristic to locate the parts. They extend the heuristic methods by allowing for some randomization. The use of packing heuristics greatly limits the size of the search space and often ensures feasibility in terms of overlap and protrusion. GA in 3D have been applied to packing cuboids in cuboids while optimizing the location of the center of gravity in the X-Y plane (Kawakami et al., 1991; Wodziak and Fadel, 1994). Extensions to cuboids with holes (shoebox-like objects with thick walls) have recently been tried (Ikonen et al. 1997). Related work in 3D pipe routing uses tesellated representations of objects to speed up the evaluations required by GA in (Sandurkar et al., 1997), allowing more complex objects to be considered.

Simulated Annealing (SA): This is a method of optimization based on the physical process of annealing metals (Kirkpatrick et al., 1983). SA is an iterative method. Solutions are subject to random perturbations. Perturbations that lead to better merit function values are always accepted. Perturbations that lead to worse values are sometimes accepted based on a time dependent "temperature" parameter and the amount of change in the merit function. If a perturbation is accepted the current solution is replaced by the perturbed solution. The proportion of worse perturbations accepted decreases with time and the algorithm evolves in a continuous fashion from behaving like a random walk to behaving like a hill climb. SA has been used extensively for layout problems in integrated circuit design. This led to the use of SA for packing 3D objects (Szykman and Cagan, 1993; Szykman and Cagan, 1995), which demonstrated that SA works well for packing cuboids and cylinders in cuboids. (Szykman, 1995) also shows examples of using SA for packing cylinders subject to connectivity costs between objects. The latest development in this methodology, (Kolli et al., 1996), is the use of multi-resolution models for fast interference detection, which reduces the time required for evaluation and facilitates the packing of non-convex objects by SA.

Other Methods: An atypical approach to packing can be found in (Kim and Gossard, 1991) where a nonlinear programming approach is used. The formulation of the problem is extremely complicated (the objects are represented as unions and intersection of half spaces) and the only example with 3D objects has only 3 objects, one of which is a cuboid. It appears unlikely that such an approach will ever be competitive with heuristic methods. Another approach which has been considered is Tabu Search. For example (Dowsland, 1993) states that it would probably work better than SA. There is, however, little empirical evidence to confirm or deny this claim.

Real world problems are usually larger in terms of the complexity of objects considered and in the number and complexity of constraints than most of the methods described above can tackle. Heuristic methods are fast, but are fragile and difficult to extend as the number and characteristics of constraints change. These methods work best with cuboids. General purpose zero order randomized methods such as SA and GA are broadly applicable but require many evaluations, usually restricting applications to fairly simple objects such as cuboids and cylinders, or to problems with a large number of symmetries.

SPATIAL LAYOUT A-TEAM

We are exploring how A-Teams can help solve complex problems. There are three main thrusts: (1) modular constraint handling for flexible organizations of custom solvers, (2) effective use of different object representations such as multi-resolution oct-trees and tesselated boundary representations and (3) combinations of different methods such as GA, ad-hoc heuristics and gradient based algorithms in a single solver. In this paper we describe the modular constraint handling thrust.

The A-Team based spatial layout tool is designed to solve problems with multiple constraints, without restrictions on the number or types of constraints. Since it is usually easier to develop algorithms or heuristics to solve problems with fewer constraints, we partitioned the problem by constraints and objectives. Independent heuristics were developed to improve solutions with respect to each constraint and objective. These heuristics were encapsulated into agents, and organized into an A-Team.

We refer to agents that work only on one or a small number of constraints and/or objectives as specialized agents since they are experts on their own constraints and/or objectives and ignorant of others. The ability of specialized agents each of which has knowledge about only one constraint (or objective) to solve problems with multiple constraints has been demonstrated in the domain of nonlinear programming in (Talukdar et al. 1997).

The hope is that such specialized agents can be reused to develop customized solvers within a class of problems with sets of overlapping constraints (and objectives), allowing the same agent to be used in several solvers, and simplifying the reconfiguration of custom solvers for new problems, where the appropriate set of specialized agents can be assembled from a repository.

Agents in A-Teams can all work in parallel across a network of computers. No coordination is required, simplyfing the assembly of the team. Should some of the agents be computationally expensive relative to the rest, several copies can be incorporated into the team to work on different solutions simultaneously.

Pareto Frontiers

Since A-Teams work with populations of solutions and do not require a single merit function, sets of non-dominated solutions (a Pareto frontier) can be constructed. As a result, the user does not have to provide a set of weights a-priori and can choose the best trade-offs between the different objectives after good solutions have been found.

The functionality of the team was extended to construct and display Pareto frontiers by adding a new agent. This agent retrieves solutions from the memory and provides plots of the objectives, allowing the user to determine solutions that are on the Pareto frontier.

An example of such a plot is shown in Figure 2. This allows the user to view the relative tradeoffs between objectives for the set of solutions found by the A-Team. A solution dominates another if its is at least as good for all objectives (and constraints) and better in at least one. A set of solutions is non-dominated if none of the solutions is dominated by another. For example, two solutions are non-dominated if one has a lower connectivity cost and the other a lower accessibility penalty.

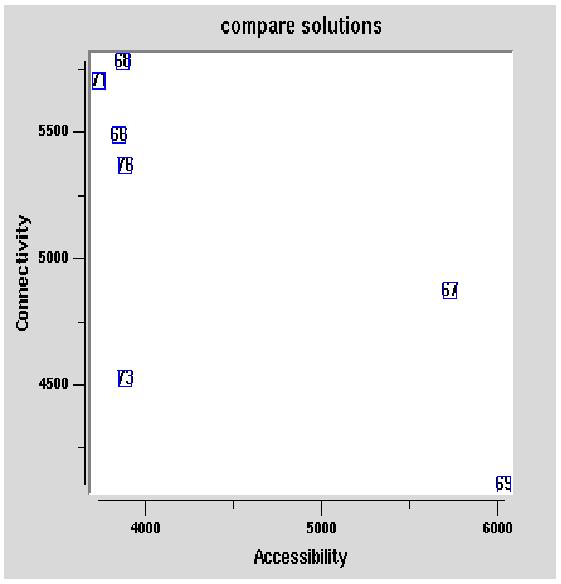

Figure 2. Pareto Frontiers: Comparing the tradeoff between objectives for solutions, each of which is identified by a unique number on the plot

Problem Definition

We selected a small set of constraints in order to develop and demonstrate the spatial layout tool. The constraints were selected to represent several different types of constraints found in practice, discontinous, continuous-differentiable, and procedural. The agents for each constraint can use the best methods for dealing with the constraint, for example gradient-based methods can be used for differentiable constraints.

Given a housing and a set of components, we specify the following constraints and objectives:

c1. Protrusion: $C_i \cap H = C_i, \forall i$.
c2. Overlap: $C_i \cap C_j = \emptyset, \forall i, j; i \neq j$
c3. Connectivity: $\min \sum_{i,j} W_{ij} \times \text{Distance}(C_i, C_j)$
c4. Separation: $\text{Distance}(C_i, C_j) \geq s_{ij}$
c5. Accessibility: $\text{SweepFace}(f) \cap C_i = \emptyset, \forall f, i$
c6. Accessibility: $\min \sum_{i,k}(\text{Volume}(\text{SweepFace}(f) \cap H)$

where:
H is a solid model for the housing packable volume (interior of the housing);
C_i is solid model for component i;
f is a (planar) face on a component that needs to be accessible;
W_{ij} is the cost of connecting components C_i and C_j;
s_{ij} is the minimum required separation between C_i and C_j;
$\text{Distance}(C_i, C_j)$ is the distance between the two components;
$\text{SweepFace}(f)$ is the solid body generated by sweeping face f, a face on one of the components, to infinity in the direc-

tion of its normal; and
Volume(C) is the volume of a solid body C

The connectivity objective (c3) measures the total cost of establishing physical connections between components. Separation constraints (c4) keep components apart that might interfere with each other electromagnetically or thermally. Certain faces on some of the components are required to be accessible (i.e. not blocked by any other component, for instance the battery compartment of a portable radio needs to be accessible). Accessibility (c5) of a face is defined as the ability to sweep a face along the direction of its normal without the swept volume intersecting with any other object. An alternate definition of accessibility (including both c5 and c6) was also used to demonstrate the reconfigurability of the A-Team based tool. The alternate problem definition measures accessibility not only by the ability to sweep the required face freely, but also by the distance of the face from the boundary of the housing packable volume measured by the intersection between the swept face volume and the housing packable volume.

Objective (c3) is designed to represent differentiable and continuous objectives. Constraint (c4) represents discontinuous constraints with many regions of zero violation. Constraint (c5) represents procedurally defined constraints. Both constraints (c4) and (c5) have highly discontinuous derivatives, which makes them difficult to solve using gradient based methods.

We use a typical electromechanical assembly design problem as an instance of the class of problems under consideration. The task consists of packing a set of components into a housing unit subject to a set of constraints. There are accessibility constraints (for components such as knobs, connectors, heat sinks etc.); connectivity requirements (for components that may have many connections between them); and separation constraints (for components that may interfere electromagnetically or thermally if placed too close together).

The artifact is the optics subassembly of a missile seeker shown in Figure 3. There are ten components in the subassembly: a ccd camera unit, a microprocessor, a digital signal processor, two analog-digital converters, a voltage stabilizer, serial and parallel connectors and two amplifiers. There are connectivity requirements between the microprocessor, digital signal processor, camera and connectors. The amplifiers must be kept suitably far from the camera and microprocessor. Additionally, the connectors must be accessible from outside the housing, and there must be no component blocking the front of the camera lens. The housing is non convex, and contains holes. The components cover a range of shapes and sizes. The shapes of the objects and constraints between them are illustrated in Figure 4

The objects are represented by 3D CAD models in ACIS format. The ACIS 3.0 class library and API are used for manipulation and processing of the 3D models. Solutions are represented as vectors of translations of the objects from their reference locations in the world coordinate system. A secondary representation uses a discretized grid, with solutions represented as a list of grid points corresponding to the centroid location of each object. ACIS 3.0 is used to evaluate the interference, protrusion, and accessibility requirements. The agent for viewing solutions uses OpenInventor for display and rendering.

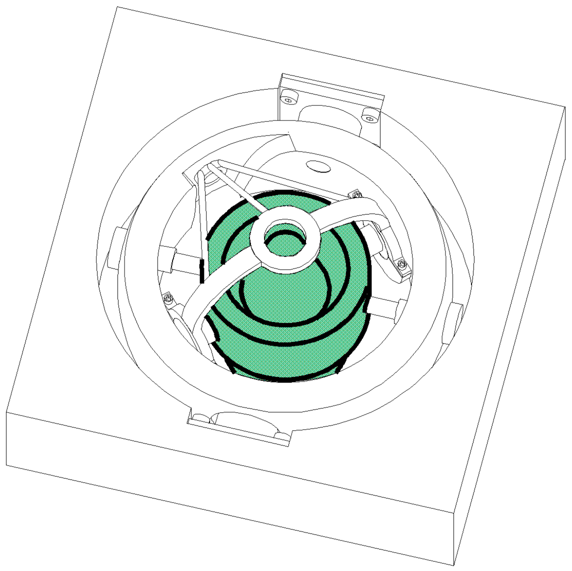
Figure 3. The Seeker Assembly for a Missile (The optics housing is colored)

The A-Team

We used two solution representations in the solver. One is a permutation list of the type used in a GA. The list identifies the order in which the objects will be placed in the housing by the decoder. The second solution representation consists of the translation of each object identifying its location explicitly in the housing.

Two types of memories (corresponding to the two representations used) and a collection of agents were developed. Such agents can be easily assembled into A-Teams resembling the example dataflow provided in Figure 1.

We categorize construction agents into three types: (1) Creators (or seeders), which generate new solutions to populate the memories; (2) Modifiers which create new solutions by modifying existing solutions in the memories; and (3) Evaluators which evaluate the solutions with respect to the objectives and constraints.

There is an independent evaluator for each constraint and objective. Evaluators start work whenever they detect

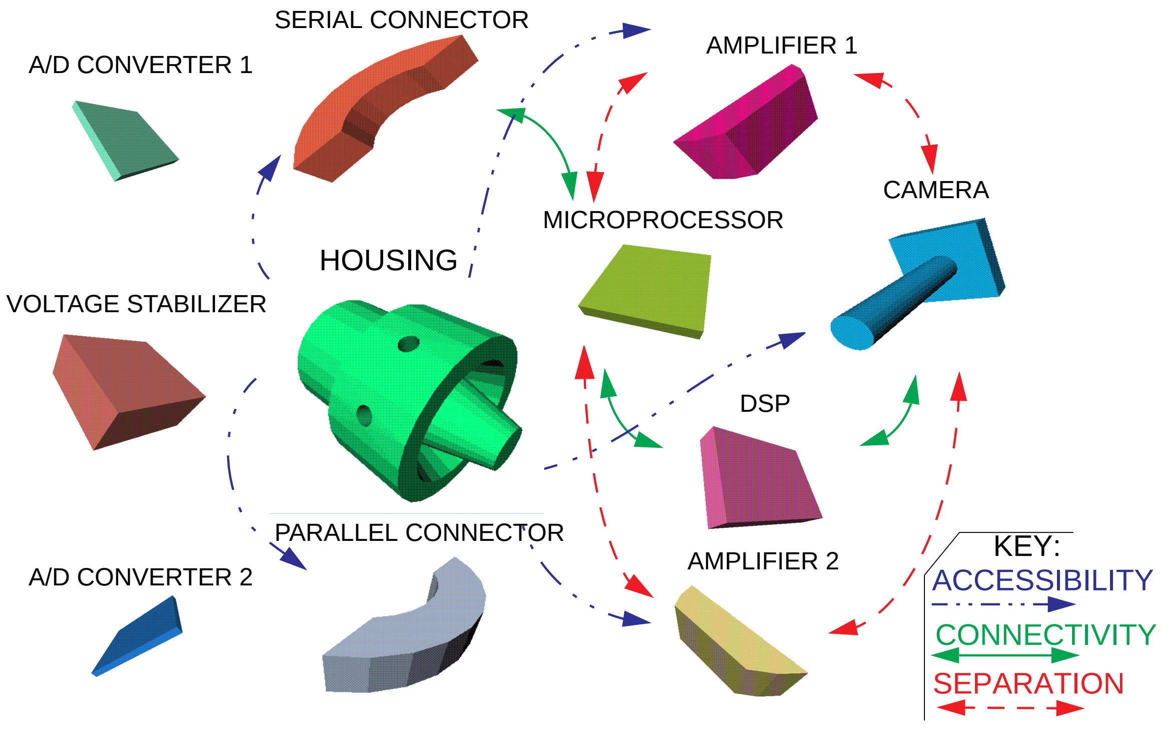

Figure 4. Problem Description: Optics Housing, Components and Constraints

new unevaluated solutions in the memory. Since they are independent, they can simultaneously update evaluations for new solutions or even work on different solutions.

There are independent modifiers for each constraint and objective. Since the protrusion and interference constraints are difficult to satisfy, agents for accessibility, separation and connectivity attempt not to increase violations of these constraints. These agents use a variety of heuristics, restricted only by the availability of good methods. As an example, one heuristic for connectivity in the A-Team uses the gradient of the connectivity cost to improve solutions. Because the modular nature of the team allows multiple agents for a constraint to run concurrently, other agents using different algorithms to improve connectivity may also work in parallel.

The current set of creators can be classified into three groups: one that uses an ordering of the objects combined with a decoder (similar to packing heuristics found in the Operations Research literature and in Genetic Algorithms), one that uses random placement of the objects (similar to that used in Simulated Annealing or Tabu Search) and one that uses a discretized grid to locate objects. We will describe one of the creators as an example of how secondary representations can be used to assist in optimization. An A-Team that uses a discretized grid representation ignores protrusion and interference, trying to place accessible objects near the grid boundary. Separation and connectivity are approximated by integral manhattan distances on the grid. The only goal of this subsidiary A-Team is to generate good starting configurations.

A destroyer erases solutions that are dominated by (and hence are worse than) others in the memory. If the memory is full of non-dominated solutions, an arbitrarily selected solution may be destroyed to make space.

Creating Memories, Agents and A-Teams using the A-Teams Toolkit

The memory is implemented as an active repository in the A-Teams toolkit. Since the agents must be able to communicate with the memory, they must share a common communication protocol. The toolkit is implemented primarily in C++. A memory stores a class of data. The user must provide a read and write method for each data class. The user specifies which class is stored in the memory using a template that is provided.

Each heuristic is encapsulated in an agent by adding a scheduler which determines how often the agent will work and the selector which determines what it will work on. In the case of the layout tool, the scheduler pauses for a pre-specified amount of time between iterations (usually about 5 microseconds). Depending on the agent, the se-

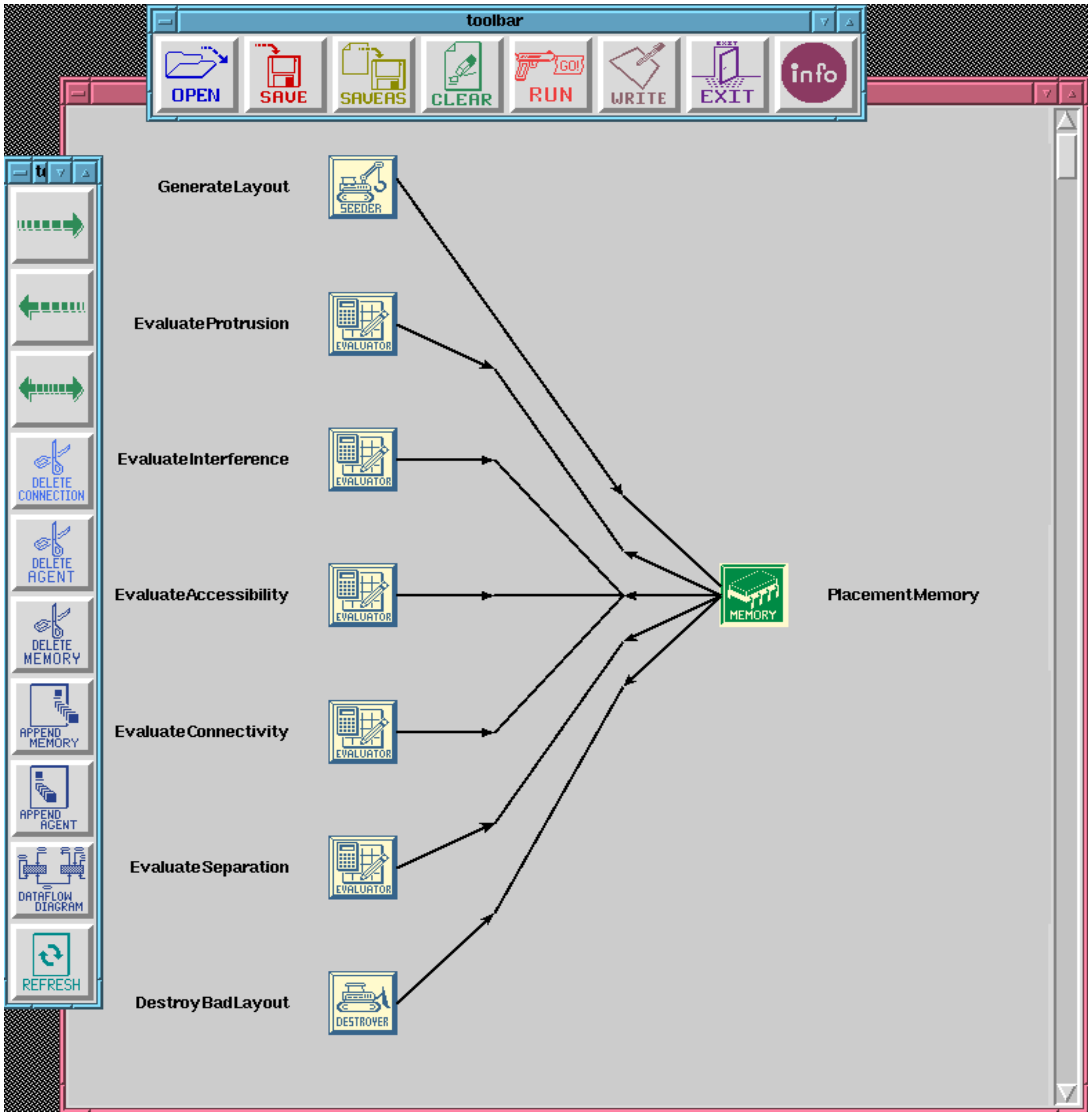

Figure 5. The A-Team Toolkit allows users to configure and execute A-Teams graphically

lector chooses a solution randomly, either from a uniform distribution (all solutions have an equal probability of being selected) or with a bias towards better solutions (e.g. those with fewer violations).

To create an agent the user must provide code for the heuristic or algorithm that is to be encapsulated. This is called the operator of the agent. The user can then either create a new scheduler and/or selector or choose one from those provided with the A-Teams toolkit.

Once the user has specified the input and output data classes, the operator, scheduler and selector can then be compiled into an agent using the template provided with the toolkit.

After the user has registered the memories and agents into a repository, A-Teams can be configured and executed using the graphical interface provided with the Toolkit. The user adds memories and agents to the A-Team, and connects them together graphically to indicate the data flow. The interface allows the user to control the execution of the team, and to add or remove agents dynamically during execution.

Tests

The tool was first used to find a placement of the objects given the accessibility, connectivity, separation, interference and protrusion constraints. It took an A-Team sharing a single sparc-ultra workstation 12 hours to find a good layout. The same A-Team took about 2 hours to find a layout of similar quality using a total of 8 networked SGI, sparc-20 and sparc-ultra workstations.

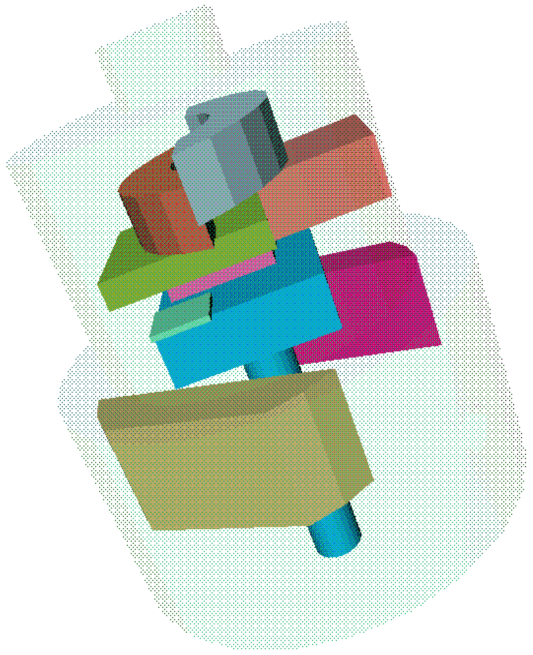

Figure 6. Layout for the problem using the first definition of Accessibility

A solution satisfying the constraints is shown in the Figure 6. The ability to customize the solver was demonstrated by using two different definitions for accessibility. It was considered insufficient for the appropriate face of the connectors to just be reachable from outside, and accessibility was redefined to give preference to placements where the connectors and the heat sinks of the amplifiers were closer to the housing boundary.

Since the A-Team was designed to be modular with respect to constraints, there is a separate agent designed to improve each constraint. Keeping a clear distinction between the sub-tasks assigned to each agents makes it easy to reconfigure the team should task requirements change.

We have shown this by modifying the specifications for accessibility, and by reconfiguring the A-Team just by replacing the accessibility related agents. The remaining agents were unchanged and, in fact could continue working while the accessibility related agents were reconfigured. A

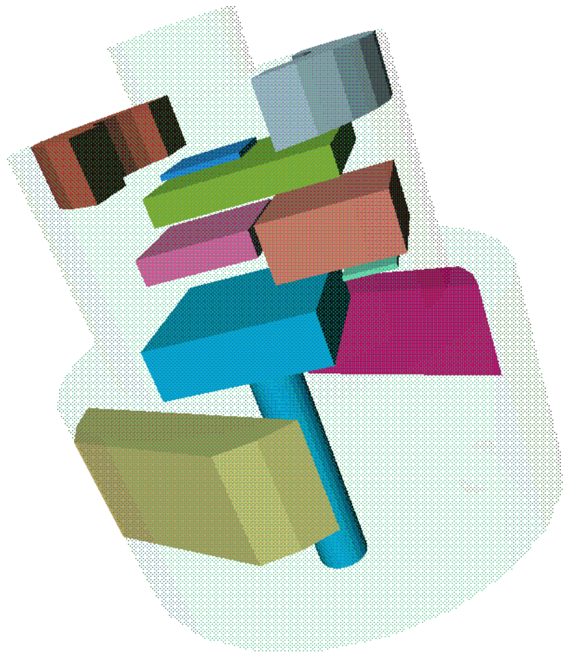

Figure 7. Layout for the problem using the second definition of Accessibility

solution found by the modified team is shown for comparison in Figure 7. Some of the components have been moved closer to the housing. Neither of the two solutions dominates - the first is better in terms of connectivity, while the second is better for accessibility. The final choice is left to the user.

SUMMARY AND FUTURE WORK

We have developed an A-Team based tool for spatial layout. The tool is modular, with separate agents for each constraint. To adapt the solution approach to a specific problem instance, the tool can be reconfigured (even during execution) by removing or adding agents to the team. We have demonstrated that a set of independent agents working in parallel on small subsets of the constraints can be organized to solve the larger problem.

We developed the tool using a set of random packing problems and tested it with a spatial layout problem representing an electromechanical assembly with 10 components, a highly non-convex housing, and 5 constraints. We are extending the test set to include more examples such as the layout of the components under the hood of a car.

The tool is also open to the use of different heuristics. It allows the exact evaluation based heuristics to be used alongside approximation based ones. We are extending the set of agents to include the use of evaluation based heuristics.

We are extending the tool to use both multi-resolution representations of objects as in (Kolli et al., 1996) and tesselated representations as in (Sandurkar et al., 1997). Both techniques appear extremely promising in initial testing, with interference detection being considerably faster for these representations and we are working towards integrating these representations into the tool.

The A-Team paradigm facilitates interaction and cooperation between humans and computational agents to solve difficult problems. This ability is exceptionally useful for Spatial Layout problems since human beings are very good at identifying and distinguishing promising from unpromising layouts. In a significant number of cases, humans are much better than the best algorithms available. Computers, on the other hand, are much better at numeric processing. We are extending the tool to allow for synergistic cooperation, harnessing the pattern recognition ability of humans with the numerical abilities of computers.

Other extensions to the tool include the use of alternate representations for objects and solutions, allowing rotational freedom, and addition of diversity (e.g. by adding a simulated annealing agent) to the team.

ACKNOWLEDGMENT

This research was funded in part by DARPA under contract ONR #N0014-96-1-0854, and by the Institute for Complex Engineered Systems and the Robotics Institute at Carnegie Mellon University.

REFERENCES

Aarts, E. , and Lenstra, J. K. (editors), *Local Search in Combinatorial Optimization*, John Wiley & Sons, New York, NY, 1997.

Aarts, E. H. L., van Laarhoven, P. J. M., Liu, C. L., and Pan, P., "VLSI layout synthesis," in (Aarts and Lenstra, 1997) Ch 12, 1997.

Aziz, N. M., "A computer-aided box stacking model for truck transport and pallets," *Computers in Industry*, v17, 1991, pp. 1-8.

Baker, K. R., *Introduction to Sequencing and Scheduling*, Wiley, NY, 1974.

Chen, C.L., "Bayesian Nets and A-Teams for Power System Fault Diagnosis," Ph. D. dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA, 1992.

Chen, S. Y., Talukdar, S. N., and Sadeh, N. M., "Job-Shop-Scheduling by a Team of Asynchronous Agents," *IJCAI-93 Workshop on Knowledge-Based Production, Scheduling and Control*, Chambery, France, 1993.

Davis, L., (editor), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

de Souza, P., "Asynchronous Organizations for Multi-Algorithm Problems," Ph. D. dissertation, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 1993.

Dowsland, K., "Some Experiments with Simulated Annealing Techniques for Packing problems," *European Journal of Operational Research*, v68, 1993, pp.389-399.

Dowsland, K. A., and Dowsland, W. B., "Packing Problems," *European Journal of Operational Research*, v.56, 1992, pp.2-14.

Gehring, H., Menschner, K., and Meyer, M., "A Computer Based Heuristic for Packing Pooled Shipment Containers," *European Journal of Operational Research*, v44, 1990, pp. 277-288.

Glover, F., "Tabu Search-Parts I and II," *ORSA Journal of Computing*, Vol. 1. No. 3, Summer 1989 and Vol. 2, No. 1, Winter 1990.

Gorti, S. R., Humair, S., Sriram, R. D., Talukdar, S., and Murthy, S., "Solving Constraint Satisfaction Problems Using A-Teams," *AI-EDAM*, vol. 10, pp 1-19, 1996.

Haessler, R. W., and Talbot, F. B., "Load Planning for shipments of low density products," *European Journal of Operational Research*, v. 44, 1990, pp. 289-299.

Hinzman, B., "The Personal Factory," *Rapid Prototyping Journal: Internet conference*. Dec. 1995. ISSN 1355 2546. http://www.mcb.co.uk/services/conferen/dec95/rapidpd/hinzmann/backgnd7.htm

Ikonen, I. T., Biles, W. E., Kumar, A., Ragade, R.K., and Wissel, J. C., "A Genetic Algorithm for Packing Three-Dimensional Non-Convex Objects Having Cavities and Holes," *Proceedings of the 7th International Conference on Genetic Algorithms*, Michigan State University, East Lansing, MI, 19-23 July, 1997.
http://www.spd.louisville.edu/ itikon01

Kawakami, T., Minagawa, M., and Kakazu, Y., "Auto Tuning of 3-D Packing Rules Using Genetic Algorithms," *IEEE/RSJ International Workshop in Intelligent Robots and Systems IROS '91*, Osaka, Japan, pp. 1319-1324.

Kim, J. J., and Gossard, D. C., "Reasoning on the Location of Components for Assembly Packaging," *Transactions of the ASME, Journal of Mechanical Design*, v. 113, December 1991, pp. 402-407.

Kirkpatrick, S., Gelatt, C.D., and Cecchi, M.P., "Optimization by Simulated Annealing," *Science*, Vol. 220, Number 4598, May, 1983.

Kolli, A., Cagan, J., and Rutenbar, R., "Packing of Generic, Three-Dimensional Components based on Multi-Resolution Modeling," *Proceedings of the 1996 ASME Design Engineering Technical Conferences*, 96-DETC/DAC-1479, Irvine, California, 1996.

Lee, H., Murthy, S., Haider, W., and Morse, D., "Primary Production Scheduling at Steel making Industries," IBM report, 1995

Lengauer, T., *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, Chichester (England), 1990.

Lukin, J. A., Gove, A. P., Talukdar, S. N., and Ho, C., "Automated Probabilistic Method for Assigning Backbone Resonances of (13C, 15N)-Labelled Proteins," *Journal of Biomolecular NMR*, 9, 1997.

Murthy, S., "Synergy in Cooperating Agents: Designing Manipulators from Task Specifications," Ph.D. dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 1992.

Quadrel, R.W., "Asynchronous Design Environments: Architecture and Behavior," Ph. D. dissertation, Department of Architecture, Carnegie Mellon University, Pittsburgh, PA, 1991.

Rachlin, J., Wu, F., Murthy, S., Talukdar, S., Sturzenbecker, M., Akkiraju, R., Fuhrer, R., Aggarwal, A., Yeh, J., Henry R., and Jayaraman, R., "Forest View: A System For Integrated Scheduling In Complex Manufacturing Domains," IBM report, 1996.

Sandurkar, S., Chen, W., and Fadel, G.W., "Gaprus: Three-Dimensional Pipe Routing using Genetic Algorithms and Tessellated Objects," *Proceedings of 1997 ASME Design Engineering Technical Conferences*, DETC97/DAC-3982, Sacramento, California, 1997.

Szykman, S., "Optimal Product Layout Using Simulated Annealing," Ph.D. Dissertation, Mechanical Engineering Department, CMU, Pittsburgh, 1995.

Szykman, S., and Cagan, J., "Automated Generation of Optimally Directed Three Dimensional Component Layouts," *Advances In Design Automation*, DE-Vol. 65-1, 1993, pp. 527-537.

Szykman, S., and Cagan, J., "A Simulated Annealing-Based Approach to Three-Dimensional Component Packing," *Transactions of the ASME*, Journal of Mechanical Design, v. 117, June 1995, pp. 308-314.

Talukdar, S., "Autonomous Cyber Agents: Rules for Collaboration," *Proceedings of the thirty-first Hawaii International Conference on System Sciences* (HICSS-31), Hawaii, January 1998.

Talukdar, S. N., Pyo, S. S., and Giras, T., "Asynchronous Procedures for Parallel Processing," *IEEE Trans. on PAS*, Vol. PAS-102, NO 11, Nov. 1983.

Talukdar, S., Baerentzen, L., Gove, A., and de Souza, P., "Asynchronous Teams: Cooperation Schemes for Autonomous Agents", to appear in *Journal of Heuristics*, 1998.

Talukdar, S.N., and deSouza, P., "Insects, Fish and Computer-Based Super-Agents," *Systems and Control Theory for Power Systems*, Chow, Lokotovic and Thomas(eds), Vol. 64 of the Institute of Mathematics and its Applications, Springer-Verlag, 1994.

Talukdar, S.N., and Ramesh, V.C., "A Parallel Global Optimization Algorithm and its Application to the CCOPF Problem," *Proceedings of the Power Industry Computer Applications Conference*, Phoenix, May, 1993.

Talukdar, S. N., Sachdev, S., and Camponagara, E., "A Collaboration Strategy for Autonomous, Highly Specialized Agents," *Proceedings of the SPIE, Symposium on Intelligent Systems & Advanced Manufacturing*, October 1997.

Tsen, C. K., "Solving Train Scheduling Problems Using A-Teams," Ph.D. dissertation, Electrical and Computer Engineering Department, CMU, Pittsburgh, 1995.

Wodziak, J. R., and Fadel, G. M., "Packing and Optimizing the Center of Gravity Location using a Genetic Algorithm," http://www.vr.clemson.edu/dmg/papers/Optpapers.html, Design Methodology Group, Clemson University, Clemson SC.