188095 S13-63
188095
48

Task Allocation Among Multiple Intelligent Robots

L. Gasser and G. Bekey
University of Southern California
Los Angeles, CA 90089-0782

U 620312S

Abstract

~~We~~ *describe the design of a decentralized mechanism for allocating assembly tasks in a multiple robot assembly workstation. Currently,* ~~our~~ the *approach focuses on distributed allocation to explore its feasibility and its potential for adaptability to changing circumstances, rather than for optimizing throughput. Individual "greedy" robots make their own local allocation decisions using both dynamic allocation policies which propagate through a network of allocation goals, and local static and dynamic constraints describing which robots are eligible for which assembly tasks. Global coherence is achieved by proper weighting of "allocation pressures" propagating through the assembly plan. Deadlock avoidance and synchronization is achieved using periodic reassessments of local allocation decisions, ageing of allocation goals, and short-term allocation locks on goals.*

1 Introduction

The coordination of several robots in a flexible assembly workstation is a problem of growing importance. Three levels of coordination are necessary:

1. *Planning*: Assembly tasks must be decomposed and represented as cooperative arrangements among several assembly robots [4].
2. *Resource Allocation*: Particular robots must be assigned individual tasks in a detailed assembly plan, in ways that assure all tasks are carried out with optimal throughput [5].
3. *Coordinated Motion Planning*: Robot effectors must be controlled dynamically as they move close together in concert, while avoiding collisions.

In this paper we are concerned only with the second coordination level - allocating tasks to robots. In a workstation of limited size, for tasks of limited complexity, we may be able to allocate robots to tasks using a centralized global allocation mechanism, for example, one based on heuristic search of the space of possible allocations. As static task or workstation complexity increases, however, a global solution becomes more costly. Moreover, if we assume that orders to the flexible assembly station arrive randomly, necessitating reconfiguration, it will be very difficult to recalculate the global allocation plan each time to "fold in" new orders with common subassemblies.

For these reasons, we have decided to explore a decentralized approach to task allocation. In this scheme, a collection of robots is *greedy* for work; they are eager to take on whatever work they can do, and as much of it as possible. Each robot makes its own decision about what task to take on, based on its own local decision criteria. The robots' greed is mediated by a set of dynamically changing *allocation policies*, which assures that global throughput requirements are met, and which encourages individual robots to assign themselves the most appropriate tasks given the global circumstances, such as order due dates, parts availability, etc.

Our focus here (and our motivation for investigating concurrency) is to explore *the feasibility of a distributed solution* and *adaptability to changing circumstances*, rather than optimal throughput, or maximum production or reallocation speed. We have not yet addressed the temporal scheduling of task assignments to achieve desired throughput results, though we shall discuss some ideas for handling temporal constraints on processing.

2 The Nature of Assembly

Assembly is the process of composing higher-level structures from parts (primitive structures) and lower-level subassemblies using *assembly operations* such as welding, fastening, screwing, riveting, inserting, etc. The assembly task allocation process must account for producing some number of copies of the high-level assemblies it generates - there is some *lot* of assemblies to make. Each lot is assembled on the basis of an *assembly plan* for an individual object in the lot, and on the basis of some *due date* for the entire lot. The due date places a time constraint on the assembly process.

Based on a preliminary analysis of several common manufactured objects, we have derived a canonical form for the high level description of coordinated assembly tasks in an assembly plan. Figure 1 shows our scheme. An assembly plan is shown as a tree-structured composition of subassembly plans. Nodes in this tree are primitive assembly operations. Assembly operations may require either single robots or groups of coordinated robots. We have only illustrated coordinated assembly tasks which require pairs of robots, but the scheme remains analogous for either single-robot tasks or those requiring more than two robots.

Any high-level assembly A is composed of several subassemblies, in this case S_1 and S_2. The assembly operation O_3 which assembles S_1 and S_2 to produce A comprises two subtasks, L_{3a} and L_{3b}, (which might be a HOLD and an INSERT operation, respectively). Each subtask has an associated *resource constraint* $R_{L_{3a}}$ and $R_{L_{3b}}$, respectively. Resource constraints have static and dynamic components. The static resource constraints describe the machines which must be used to perform the subtask, and may indicate the type of tooling required, the lifting capacity or application force required, etc. Dynamic resource constraints describe constraints which vary with the allocation process (such as the real physical proximity of the required partner robot), once some robot has has taken on one task of the pair. The same description notation is used at each level of the assembly plan. Lowest-level subassemblies are made from *parts* (P).

2.1 Allocation Pressure

The existence of any unfulfilled task in the assembly plan creates two subgoals: a global *goal* of allocating some robot to the task and a local *performance goal* goal of performing the task once it has been allocated. However, we use a decentralized approach to allocation, meaning that the individual decisions about which robots respond to which goals are taken by the robots individually, not by a global allocator. Still, to maintain global coherence, the global task-allocation goals must be ordered to reflect global priorities. We do this by establishing dynamic local *policies* to guide individual allocation decisions. These policies are not explicit decision rules; instead they are weighting factors attached to each allocation goal indicating the global importance of the goal. The weights are called *allocation pressures* for the goals. A higher allocation pressure makes a given goal more attractive to any robot. The combination of allocation pressures, resource constraints, and the opportunistic decisions of individual robots controls the allocation process over time.

Allocation pressures attached to goals are dynamic and local. They change over time as some tasks are completed and others become more pressing. They are created by propagating allocation pressures through the assembly plan. Allocation pressures come from three sources, and are termed *production pressure*, *coordination pressure*, and *consumption pressure*. First, lower-level subassemblies must be created before higher-level ones can be assembled; this places a precedence on the order of assembly, and thus establishes some precedence for task allocation (e.g., when there are fewer robots than assembly tasks, or some robots are faster, or better-suited for certain tasks than others). As each assembly order arrives, it carries a *due date*. The due date for the completed lot provides the top-level *production pressure* which propagates downward (i.e. toward finer-grained subassemblies) with increasing force. This serves to encourage robots to choose lower in the assembly plan at first, since it would make no sense to take on higher-level assembly tasks if there were nothing to assemble. As the due date approaches, the production pressure *increases*, ageing the allocation goals over time.

Second, if one robot (X) decides to assume some subtask L_{ia}, which is part of a coordinated task O_i, it must be sure that it can induce some other robot (say, Y) to take on the corresponding subtask L_{ib}, in order to proceed. We call this the *principle of complementarity in allocation*. As one robot makes a decision to assume a task, it creates 1) a new set of dynamic resource constraints on the corresponding subtask (because the allocation of one robot determines the physical location where the task will occur), and 2) creates *coordination pressure* to encourage some available and appropriate robot to assume the complementary subtask (since without a partner, the operation is not possible).

Third, the production of lower-level assemblies creates an upward-propagating *consumption pressure*. This encourages some robots to take on the higher-level assembly tasks in order that work-in-process inventories (e.g. parts storage bins) and work flows remain balanced. Consumption pressure increases as more lower-level subassemblies are produced. It is initiated at the parts level, by arrival or availability of parts at the workstation.

Allocation pressures are propagated using weighted propagation functions on the arcs of the assembly plan. The precise nature of the propagation functions is to be determined by experimentation and theoretical analysis, which await further research.

2.2 Local Allocator Decisionmaking Criteria

Individual robots make local allocation decisions by examining the available unsatisfied allocation goals and selecting those with compatible resource constraints. From this set, the most highly-rated allocation goal is selected, and the robot allocates itself (see implementation section below). This allocation decisionmaking takes place with some fixed periodicity, as well as happening any time a robot becomes available for work. It is important to have repeated, periodic checks, for adaptive allocation and deadlock avoidance.

If a robot has no current task (i.e. it is free to take on any compatible task) then it may decide purely on the basis of compatibility, However, if the robot is engaged, yet is doing a periodic re-allocation check, it must also consider the *changeover cost* in deciding whether to take on a new task.

Changeover costs include the costs of retooling, opportunity costs for not getting the new work done, and the prevailing performance goals for the task it is already performing.

These decisionmaking criteria are purely local, both with respect to the individual tasks, and to the temporal unfolding of the global assembly work. It is possible that, for maximum throughput, an individual robot should avoid taking on a task which is immediately available, and should wait for an upcoming but currently lower-rated task to which it is better suited. If the upcoming task is one in the current set of globally-known allocation goals, this requires each robot to incorporate in its decisionmaking criteria either 1) meta-level control policies which can be constructed by some planner with a more global view or by integrating information about what other robots are doing by communicating with them [1], or 2) some predictive knowledge about the expected trajectory of the allocation pressures through the system. If the upcoming allocation goal will be generated by an assembly order which itself has not been generated, the individual robot will need even higher-level information about the likely arrival of new assembly orders.

With concurrent access to all allocation goals in an assembly plan, there is a potential for deadlock. If two robots with conflicting reach constraints simultaneously choose complementary subtasks, (e.g. L_{ia} and L_{ib} of task O_i) the operation cannot proceed, but there will be no way to deallocate any robot. Moreover, there is no criteria for deciding which robot should be deallocated. This is only a problem for subtasks of the same operation, because they must have *simultaneous allocation* of appropriate resources. It is not a problem for subtasks of different operations, which can be allocated concurrently. Inappropriate allocation of robots across different operations will lead to inefficiency, but the adaptation mechanisms provided by the propagating allocation pressures, ageing of goals with respect to due dates, and the re-assessments of allocation decisions will force the system to correct itself.

We use two mechanisms to prevent deadlock among subtask allocations. First, allocation access to all subtasks of a single assembly operation must be locked so that only one allocator at a time can make an allocation decision. After this decision is made, the new dynamic constraints posted on the complementary allocation goals assure that only appropriate partners choose the complementary subtasks. This locking, while it reduces concurrency, does not create undue overhead, because the allocation decisions are made infrequently, and are short by comparison to the amount of time it takes for actual assembly operations.

Second, if no partner chooses the complementary subtask within the time constraint provided by the allocation reassessment period, the allocation goal's value may change, and the robot is free to deallocate itself and take on a more highly-rated goal (given the changeover cost - see below).

3 Implementation Approach

The allocation system described here has not yet been implemented, but here we present our design for implementation. The implementation is planned for MACE [2,3], our concurrent Distributed AI testbed. The basic structure of the system is designed to be a global but distributed blackboard system [3]. The blackboard itself contains the overall assembly plan and propagation links, and a collection of concurrently-executing allocation decisionmakers, one associated with each robot in the workstation.

3.1 Representation of the Assembly Plan

The assembly plan is represented as a collection of goals on a globally accessible but distributed blackboard. The blackboard may be pre-segmented according to machine types and allocation constraints, to provide some efficiency in allocation and communication, and to allow for increased concurrency as individual allocators access different parts of the blackboard [3]. Each allocation goal in the assembly plan is represented as a collection of constraints on the type of robot which can assume the task. Each goal has two constraint sections: a dynamic and a static part. Constraints are descriptions of robot characteristics expressed in a flexible pattern-language [2], to allow for partial matches, restricted matches, and variable bindings for the allocators. Static constraints are fixed parts of the assembly plan, whereas dynamic constraints are updated as allocation decisions are made. The updating is done by a MACE computational agent which manages blackboard access. Allocation locks are implemented using the mailbox synchronization provided by MACE, and are controlled by the blackboard level manager.

Allocation goals are linked to one another with any of four types of uni-directional links. Each link comprises a *type* and a *propagation function* for propagating allocation pressures or constraints. The four types of links are *production-pressure links*, *consumption-pressure links*, *coordination-pressure links*, and *constraint-propagation links*. Constraint-propagation links connect subtasks of a single assembly operation, and describe how to update dynamic allocation constraints.

3.2 Local Allocation Decision-Making Agents

Local decisionmaking agents are individual MCAE agents which access the global blackboard using messages and demons. MACE provides a facility for remote demons, so that when a goal for which a particular robot is particularly well-suited is posted or its evaluation changes, the robot can be notified opportunistically. Direct access to allocation goals is achieved with messages. Local decisionmaking criteria are built into rules within each allocator agent.

4 Conclusions

We have presented the design of a promising scheme for decentralized allocation of tasks in a multiple robot assembly workstation. Further research remains on the theoretical analysis of weights and feedback through the system. This analysis should include a control-theoretic analysis of the system, to show the weighting and constraint conditions under which it is possible to assure allocations will occur with maximum throughput, and to indicate the problematic condtitions under which the system can avoid thrashing, or at least damp it. After implementation, we also expect experimental analysis to derive appropriate weights, reallocation periodicity, and adaptive performance behavior.

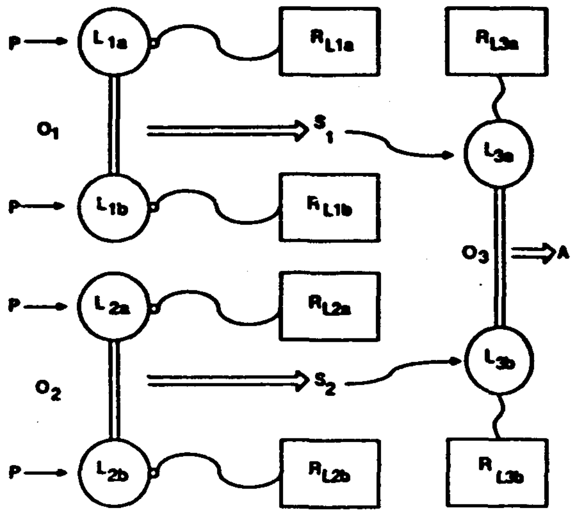

Figure 1.

References

[1] Durfee, E., Lesser, V. and Corkill, D. "Coherent Cooperation Among Communicating Problem-Solvers." *IEEE Transactions on Computers* (to appear) 1987.

[2] Gasser, L. Braganza, C., and Herman, N. "MACE: A Flexible Testbed for Distributed AI Research," to appear in M. Huhns, Ed., *Distributed Artificial Intelligence*, Pitman Publishers, 1987.

[3] Gasser, L. Braganza, C., and Herman, N. "Implementing Distributed AI Systems Using MACE" in *Proceedings of the 3rd IEEE Conference on Artificial Intelligence Applications*, Orlando, Fla., February, 1987 (to appear).

[4] K. Konolige and N. J. Nilsson. "Multi-Agent Planning." in *Proc. National Conference on AI*, 1980.

[5] Sadeh, N. and Gasser, L. "Hierarchical Scheduling and Resource Allocation," Paper submitted to the *International Joint Conference on Artificial Intelligence*, Aug. 1987.