Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2004), *18*, 135–152. Printed in the USA.

DOI: 10.1017/S0890060404040107

Interagent ties in team-based computational configuration design

JESSE T. OLSON AND JONATHAN CAGAN

Computational Design Lab, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA

(RECEIVED May 30, 2003; ACCEPTED June 28, 2004)

Abstract

Organizational research has shown that effectively structuring the resources (human, informational, computational) available to an organization can significantly improve its collective computational capacity. Central to this improved capacity is the manner in which the organization's member agents are related. This study is an initial investigation into the role and potential of interagent ties in computational teaming. A computational team-based model, designed to more fully integrate agent ties, is created and presented. It is applied to a bulk manufacturing process-planning problem and its performance compared against a previously tested agent-based algorithm without these agent relationships. The performance of the new agent method showed significant improvement over the previous method: improving solution quality 280% and increasing solution identification per unit time an entire order of magnitude. A statistical examination of the new algorithm confirms that agent interdependencies are the strongest and most consistent performance effects leading to the observed improvements. This study illustrates that the interagent ties associated with team collaboration can be a highly effective method of improving computational design performance, and the results are promising indications that the application of organization constructs within a computational context may significantly improve computational problem solving.

Keywords: Agents; Collaboration; Computational Synthesis; Configuration Design; Teams

1. INTRODUCTION

The success of a product is largely influenced by the early stages of engineering design, when the concept configuration and framework are established. At this early stage of the design process, computational aids could potentially assist the designer in achieving a more comprehensive search and evaluation of potential solutions. Design problems at this early stage, however, represent a highly demanding class of computational problems. They are often multimodal, dynamic, and open-ended. The significant variables are often coupled and the extent of their relationships unknown. Although computational tools have shown success in assisting in this early design process (Antonsson & Cagan, 2001), highly effective human teams have shown many unique and positive attributes particularly adept at taking into account the challenges of the early design process. Our hypothesis is that more effective computational teaming strategies that take advantage of human organizational understanding would lead to higher quality automated design processes and solutions. The intent of this study is to investigate computational methodologies that may more effectively account for these task complexities by capturing key characteristics of effective human design teams. This is an initial investigation of this study, focusing on a central aspect of highly effective team collaboration: interagent ties.

This paper begins with a brief review of pertinent organizational observations in Section 2. This review explores emerging organizational insights into interagent ties and the key role they play in collaborative problem solving. How these principles have been typically approached in computational modeling, and their potential benefits to computational design, are also briefly reviewed. To investigate these organizational principles within a computational context, a multiagent approach is taken in this study. Multi-

Reprint requests to: Jonathan Cagan, Computational Design Lab, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA. E-mail: cagan@cmu.edu

agent programming is an emerging technology that has shown great promise in addressing complex, dynamic problems, and with clear analogous similarities to human teaming and organization. Section 3 reviews two specific agent-based design approaches that provide an important base for this work. Additional details of one of these algorithms and the design task are provided in Sections 4 and 5. Then, Section 6 presents our computational approach, an extension of the previously discussed model designed specifically to more fully account for the organizational principles in question. Both algorithms are tested and analyses of their performance are given in Section 7 in the form of performance comparisons and a statistical study evaluating key algorithm components. Finally, in Section 8, some final discussion concerning the performance evaluation and implications are given.

2. ORGANIZATIONAL PERSPECTIVE: MOTIVATION

Organizational studies are leading to a new understanding of the nature of organizations. According to Carley and Hill (2001), organizational research has often focused on how either the organizational *structure* or the individuals' *attributes* influence organizational performance. The one has been considered independent of the potential coupling influence of the other. This separable approach has largely shaped our thinking about organizations. Today, however, organizations are increasingly viewed as complex systems in which the interrelationship between the structure and individuals' attributes is key (Carley & Hill, 2001). This view provides a new perspective on the structures and strategies most conducive to highly effective teaming, and central to these are the type and degree of interrelationships between agents (Carley, 2001*b*; Carley & Hill, 2001).

This new understanding of organizations indicates that performance is more than providing each individual its necessary part of information, resources, and capabilities to complete its task. It is also significantly dependent upon the interrelationships that link agents' knowledge, computational capability, decisions, tasks, and resources (Bar-Yam, 1997, 2001; Carley & Gasser, 1999). They entail meshing activities, sharing information, and correlating decisions (Wholey et al., 1995).

Organizational flexibility, robustness to uncertainty, and computational capacity are all understood to arise from ongoing agent interactions and interdependencies (Carley & Gasser, 1999). There is, therefore, a relationship between individuals' interdependence and the organization's collective performance. This relationship takes the form of a trade-off between large-scale behavior and fine-scale interdependencies, or interagent ties (Bar-Yam, 2001). The richer the set of fine-scale interdependencies, in brief, the more complex the organization's potential behavior (Bar-Yam, 1997, 2001).

Thus, appropriately coupling agents' knowledge, decisions, and activities may result in an increase in what we term the *collective potential*. We define an organization's collective potential as the set of collective behaviors and actions an organization is capable of producing. For organizations in which the primary task is to resolve problems through the manipulation and synthesis of information (such as the design algorithms in which this investigation is interested), the greater the collective potential, the more complex the informational structures the organization can successfully synthesize. Collective potentials greater than and unique from organizations' individual members have been observed in some highly effective organizations, and this increase is understood to arise from these fine scale interdependencies.

2.1. Organization in computational systems: Brief overview

2.1.1. Organizational sciences

The context and purpose of studies within the organizational sciences are typically to better understand human organization behavior. Nevertheless, these studies and the underlying theory do indicate that they may pertain in some degree within a computational framework as well. In fact, much of this emerging perspective on organizations is rooted in computational research. In recent years, computational modeling has become one of the primary methodologies of choice in the organization science community, and these computational organization studies have established the modelability of organizational attributes within a computational framework (Carley & Gasser, 1999).

Still, it is unclear exactly what role these principles may play and their potential influence in the context of computational performance. As the intent of organizational theory is to study and model human organizations, computational models are often intentionally designed with suboptimal characteristics to better approximate the targeted organizational attributes. Studies in this field have not taken place with the purpose of identifying the principles governing effective computational performance.

2.1.2. Distributed artificial intelligence

Correspondingly, organizational effects are rarely taken into account in distributed computational studies, in which the goal *is* generally to identify the principles governing effective computational performance. That is, they rarely investigate the potential influences of emergent, group behavior on computational performance. Rather, agents (their computational capacities, decisions, knowledge), while coordinated, are often viewed as independent and separable (Carley et al., 1992). This often occurs as a result of considering the organizations' attributes and its structure separately. The task is decomposed and distributed, resources are streamlined through communication cost functions, and agents are

defined such that they become individually more robust and efficient. The result is an increasingly independent and uncoupled group of agents, minimizing emergent group properties.

Given the specific objective and/or context of a study, it may be the case that an independent and separable structure is appropriate. For example, groups situated in a competitive setting (dissimilar or competing goals) or addressing well-defined, decomposable tasks might be most appropriately and efficiently addressed within a separable view. Nevertheless, it is pertinent to note that it is to this separable view of organizational design, even in collaborative systems in which the potential impact and benefits of structural relations are most pertinent, that studies in distributed computation overwhelmingly ascribe (Carley & Gasser, 1999). The result is that the role and potential of these organizational effects in computational systems remain largely uninvestigated.

2.2. Potential benefits of an organizational approach

Thus, the organizational approach to effective teaming of most multiagent systems is fundamentally different than that established in organizational studies. According to these organizational studies, these different approaches, separable versus interdependent, yield very different collective behaviors and performance potentials. Making agents separable and independent is to model the system as, simply, a set of decomposed parts, the parts being sparsely and essentially linearly interconnected. The resultant group performance of such an organization is effectively bounded by a simple aggregate of the participating agents' performances (Carley, 2003). Appropriately coupling agent tasks, knowledge, decisions, and capabilities, on the other hand, has been shown to lead to collective performance greater than and unique from those of the individuals (Carley, 2001*a*, 2003).

The aim of this study is to conduct an initial investigation into the role and potential that agent interdependencies may have in a computational design context. Emphasis will be placed on whether or not collective behaviors and potentials similar to those observed in human organizations can be elicited within a computational context. If they can be, the impact on performance could be significant.

3. BACKGROUND: AGENT-BASED DESIGN ALGORITHMS

The computational foundation of this work is primarily based on two related agent-based design algorithms. Both approaches take a distributed yet separable approach to design, using distribution as a method to incorporate goal-driven design decisions within a stochastic framework. These methods have both shown success in solving complex conceptual design problems.

The first, which is the A-design methodology of Campbell, Cagan, and Kotovsky (1999, 2000, 2003), is an agent-based computational design methodology aimed at automating the conceptual design process. The method merges stochastic optimization techniques, primarily genetic algorithms, with knowledge-based search to generate electromechanical design configurations. In that method, a population of solutions is maintained, and it is the basis from which new solutions are formed and compared. This population aspect of the approach differs from traditional genetic algorithms in that modifications of the designs are explicitly goal directed, and heavily influenced by deterministic knowledge about the problem domain.

In the A-design methodology, configuration agents, defined in that study as knowledge-based strategies, are combined in a cooperative, stochastic setting. The study maintains that with many agents of differing strategies sharing design responsibilities, the system is more capable of resolving complex problems than similar strategies acting in series. To ensure that the deterministic nature of individual agent strategies does not cause the system to stagnate at local minima, the agents that contribute to each design are probabilistically invoked. Although the agents in the A-design approach work together to complete design solutions, as a result of this probabilistic approach to the agent organization and the individual focus of agent knowledge and decisions, the resulting agent organization is a separable one.

A-design provided the foundation for the second algorithm, Deshpande and Cagan's (2003) work on manufacturing process planning optimization. In that work, a similar population of solutions is stored and modified by agents that cooperate within a stochastic framework. The agents are a modification of the move sets typically used in simulated annealing, having the functionality of move sets but also domain knowledge to probabilistically modify a given design state.

Deshpande and Cagan's work (2003) incorporates both process configuration and its corresponding parameter optimization. Two types of agents perform these design alterations: configuration agents and parameter (instantiation) agents. The configuration agents modify the design sequence by probabilistically adding or deleting manufacturing processes using a simple constraint satisfaction method to ensure the feasibility of all design sequences. Parameter agents modify the parameters associated with the selected processes by probabilistically applying domain knowledge specific to each process.

Both these algorithms were successful relative to traditional methods. Implementing a distributed system methodology, in these algorithms, enabled the effective use of goal-driven and knowledge-based problem solving strategies while maintaining system robustness through randomness. Both systems, though, maintain a linear, separable view of agent organization as agents' knowledge, decisions, and capacities are uncoupled and independent. The

study in this paper extends Deshpande and Cagan's work (2003) using the manufacturing process planning problem as an application. Their algorithm is redesigned, in our approach, to more fully include agent interdependencies. This is done by replacing the probabilistic calling structure and individual decision making of the original algorithm with a team-based process and structure that includes shared decision making and information sharing.

4. THE TASK: BULK MANUFACTURING PROCESS PLANS

The task addressed in this study is the design of bulk manufacturing process plans, the same task undertaken in Deshpande and Cagan (2003). This task is a serial configuration problem; each process plan is created by configuring a series of manufacturing processes together and specifying each of their operating parameters. The aim is the manufacture of a specified part at minimum cost. The input consists of the geometric and material properties for the part to be manufactured. The output consists of the set and order of the processes, and their associated parameters.

In these tests, there are seven manufacturing processes: casting, upset forging, blocker forging, preform machining, rough machining, finish machining, and extrusion. Each process can be arranged within the plan sequence in any combination, excluding or repeating any process if desired. For example, a process sequence may be casting → upset forging → upset forging → rough machining → finish machining.

Each of these manufacturing processes, then, has a range of pertinent, associated parameters that determine the exact performance characteristics of the process. The parameters associated with each process are listed in Table 1.

The overall cost of the process plan is a function of multiple factors. They include the cost of the each process with its associated parameters, run costs, performance penalty costs, and miscellaneous costs such as necessary evaluation or necessary heat treatment costs. Note that the process and parameter descriptions as well as the final evaluation functions remain unchanged from those used in Deshpande and Cagan's algorithm so as to give an accurate performance comparison. In addition, the final evaluation functions used are those modified from Fischer et al. (1997) and Gunasekera et al. (1996), as discussed in Deshpande and Cagan (2003), to allow repeating processes within the plan sequence.

Table 1. *Manufacturing processes and their associated parameters*

Processes			
Casting	Upset forging Blocker forging	Preform machining Rough machining Finish machining	Extrusion
Parameters			
Billet radius Billet height	Die geometry Die temperature Die friction	Geometry	Die geometry Die friction Die length Die temperature Die velocity

5. ORIGINAL AGENT-BASED ALGORITHM

As Deshpande and Cagan's algorithm will be used to benchmark the performance implications of our team-based approach, their algorithm and its main processes are now described. Their program follows a genetic algorithm type approach where solutions are maintained in a list and iteratively evolved over time. The process is commenced by randomly generating an initial population of designs. Then the iterative cycle begins as follows.

The population of designs is sorted. Inferior designs (here the lower 50%) are pruned away. Each of the remaining designs is, one by one, copied, and some of the parameters of the processes associated with that design are altered. The new design is evaluated and, if improved, is inserted into the final list. If the altered design is not improved, however, the configuration agents modify its process configuration. The alterations are then instantiated with process parameters. The newly altered design is evaluated and placed in the final list. Once each design has been altered, the cycle begins anew and continues until the prespecified number of iterations has been reached. A schematic overview of the algorithm is presented in Figure 1.

The parameter modifications of the cycle are performed by instantiation agents. The type, location, and extent of these alterations are, in that algorithm, all probabilistically determined. These probabilities are based on a number of techniques. For example, agent selection probabilities are dynamically adjusted in a Hustin move set (Hustin & Sangiovanni–Vicentelli, 1987) based procedure, where the number and size of modifications decrease over time and with improved design quality. Positive and negative feedback criteria are also employed to inform and adjust these probabilities. Finally, these alterations are also influenced by general guidelines based on domain specific heuristics.

If unimproved, the design advances to a configuration modification cycle, as described above. Here, in a similar fashion to the parameter alteration process, one or more configuration agents are chosen to alter the given design. In order, the chosen agents will each, individually, insert their processes into the sequence. The choice of location is also probabilistic but restricted to feasible locations through a simple constraint satisfaction criterion based on prespecified domain knowledge. If these modifications fail to produce a feasible design sequence, the configuration agents continue to modify the sequence through process additions until it becomes feasible. If the sequence exceeds a predeter-

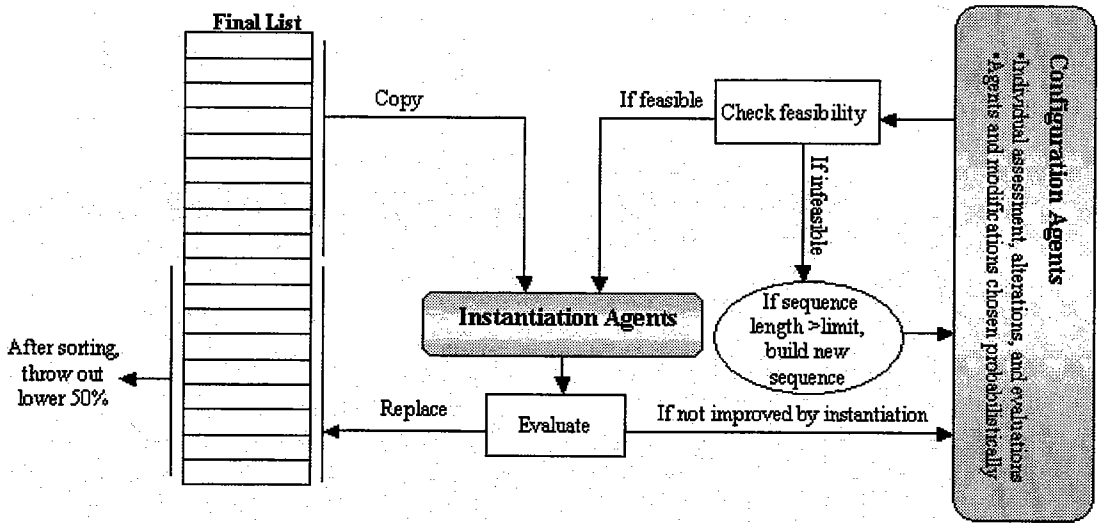

Fig. 1. A schematic overview of the Deshpande and Cagan (2003) algorithm.

mined maximum sequence length, in their implementation nine manufacturing processes, it is reset to a minimum sequence length, casting → finish machining, and the process addition cycle is recommenced. Then, once feasibly modified, the new processes are reinstantiated and the cycle is completed according to the process outlined above. Thus, modifications are made in a distributed fashion, having multiple semiautonomous contributors. Yet agents are invoked probabilistically one at a time and their modification moves are made by predetermined domain specific heuristics and individual probabilities, so the agents work independently and separable from one another. See Figure 2 for a high-level overview of the algorithm.

Finally, a few minor alterations were made to the original form of Deshpande and Cagan's (2003) algorithm so that it and the new team-based algorithm could be more accurately compared. The alterations were made to ensure that the iteration count criteria were equivalent and to ensure that the two programs did not count repeated design sequences in the final list.

```
Randomly generate initial population (both processes and parameters) & set up processes
Evaluate designs
while (# of iterations < max.iterations)
        Sort the population of designs
        Reject the poor (lower 50%)
        i = 0
        while (i < number of remaining (top 50%) designs)
                Copy design_i to location (i + population)
                Probabilistically modify parameters of each copied design
                Evaluate
                if no improvement in objective function
                        Probabilistically modify sequence configuration: choose add or subtract, the modification
                          location, and the manufacturing process
                        if resulting sequence is physically infeasible or it is a repeated design
                                while infeasible or repeated:
                                        if (sequence length > limit)
                                                Set sequence to: Casting → Finish Machining
                                        Probabilistically add processes at the location following the last change
                        Probabilistically assign parameters
                        Evaluate
                i = i + 1
        Update probabilities
```

Fig. 2. A high-level overview of the original algorithm.

6. NEW COLLABORATIVE TEAM-BASED ALGORITHM

6.1. Key algorithm extensions

The new team-based algorithm is built upon the foundation of Deshpande and Cagan's (2003) algorithm. The application in this paper addresses the same task using the same cost functions for final, complete solution evaluation. Parameter alterations are made using the same probabilistic and heuristic functions. Furthermore, the process the new algorithm follows is also iterative, evolving designs through the contribution of several semiautonomous contributors and maintaining a list of the best final designs. Both algorithms were implemented on the same, single CPU. The agents and their organization were redesigned, however, to more closely approximate human team processes and better include member interdependencies. To accomplish this there were three critical extensions made to the original algorithm of Deshpande and Cagan.

6.1.1. Extension 1: Shared action

The first key extension is the incorporation of shared action. In the new team-based algorithm, agents' activities are meshed and coupled. That is, rather than agents being invoked probabilistically and working individually as in the original algorithm, modification decisions and solution evaluations are shared and coupled between agents. Sequence modifications are collaboratively determined through a process in which agents search for agreement between individual modification suggestions and then construct collective solution modifications based on that search. So agents share information and then merge that information to derive group-based designs. Evaluations of incomplete solutions are also made as a combination of the individual agents' evaluations. Here, a set of agents individually evaluate a portion (with some overlap) of these solutions and then, again, merge their individual evaluations for a complete, group evaluation of the solution sequence. Thus, agents' actions, their decisions and evaluations, are interdependent.

6.1.2. Extension 2: Novel knowledge structure

The second extension is the introduction of a novel knowledge structure. In the original algorithm, agent knowledge is based on dynamic probabilities and prespecified domain specific heuristics, as described in the previous section. In the new algorithm, configuration agent knowledge and reasoning is based on positional relationships. To accomplish this, each agent's knowledge, decision, and evaluation schemes are centered on a particular manufacturing process, one agent per each of the seven processes. Agents then maintain evaluations of each manufacturing process (including "its" process) based on its proximity to the agent's process. Thus, each agent maintains an estimation of the value of each process located one (adjacent), two, and three or more processes from the location of "its" process in any given configuration, both to the left and right. For example, for a process addition modification, given the design sequence of Figure 3, the "upset forging" agent may determine the "best" process addition by determining the perceived value of each manufacturing process one space to its left, one space to its right, and two spaces to its right (see Fig. 3).

These agent knowledge schemes were designed to enable and encourage knowledge interrelationships, and to ensure those relations were correlated to those of the task. Because each agent evaluates solutions and modifications according to "its" process' location, from its unique perspective, its knowledge and design evaluations are unique. By the same token, because each agent evaluates solutions and modifications from the same configurations, its knowledge and evaluations are interrelated with those of the other member agents (see Fig. 4). Also, as discussed in the next subsection, this relational knowledge approach allows for the evaluation of incomplete design solutions.

Each agent's knowledge scheme consists of three elements. They are quality, confidence, and frequency. Quality

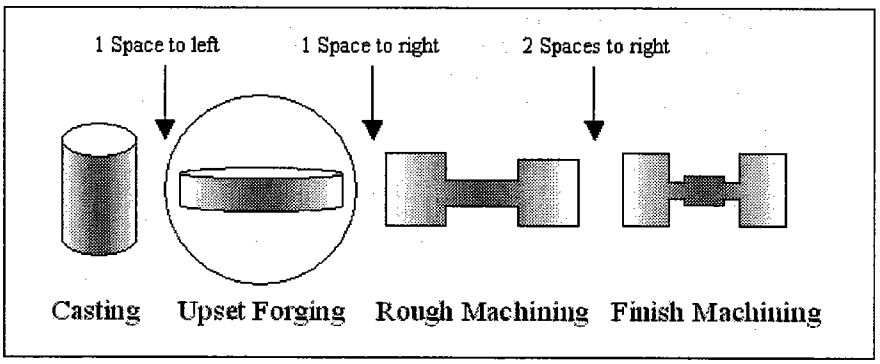

Fig. 3. An example of knowledge architecture based on spatial relationships.

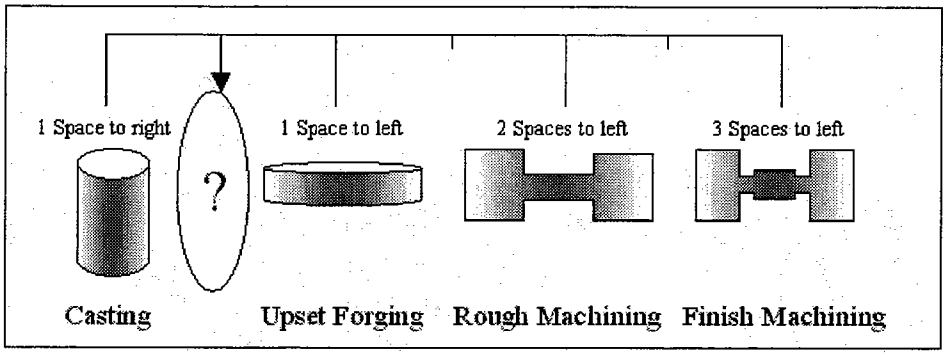

Fig. 4. A schematic demonstrating how agents consider the same modification location from a positionally different perspective.

is a measure of the estimated resultant objective function of a design containing the sequence combination in question. Confidence is a measure of the level of certainty of the indicated quality. Frequency is a measure of the number of times the sequence combination has been tried.

The three elements are all based on learning from analysis of completed design solutions. Note that the three knowledge elements are each functions of both the manufacturing process and its location relative to the agent's position within the sequence. When a design is completed, each agent examines the design and updates all three of the knowledge elements for the appropriate process/position relationships relative to the design's objective function value.

The three elements are maintained and updated according to the following equations:

$$\text{quality}_{\text{new}} = \frac{(\text{quality} \times \text{frequency}) + \text{value}}{\text{frequency} + 1}, \tag{1}$$

$$\text{confidence}_{\text{new}} = \frac{(\text{confidence} \times \text{frequency}) + \dfrac{(\text{value} - \text{confidence} + 1)}{2}}{\text{frequency} + 1}, \tag{2}$$

$$\text{frequency}_{\text{new}} = \text{frequency} + 1, \tag{3}$$

where "value" is an inverse normalization of the objective function, modified such that its range is $0 < \text{value} < 1$, a value of 1 corresponding to an objective function of zero and a value of zero corresponding to the objective function limit of 1,000,000. The value derived from the objective function and the projected values of incomplete solutions correspond one to another so that they may be directly compared and sorted.

When agent decisions and evaluations are made respective to their positional knowledge, these three elements are used as the agents' criteria with which to assign sequence combinations (in both solution modifications and evaluations) their projected value. The evaluations are made according to

$$\text{projected\_value} = \frac{\left\lfloor \sin\left(\text{quality} \times \pi - \frac{\pi}{2}\right) + 1 \right\rfloor + \left\lfloor \sin\left(\text{confidence} \times \pi - \frac{\pi}{2}\right) + 1 \right\rfloor \Big/ 2 + \left\lfloor \cos\left(\frac{\text{frequency} \times \pi}{\text{total\_solutions}}\right) + 1 \right\rfloor}{5}. \tag{4}$$

The trigonometric functions were used simply to weight the values of quality, frequency, and confidence as they approached 1 or zero more heavily. Other functions, such as a simple linear function, could just as appropriately be applied.

Note that the quality and confidence variables are convergent, and frequency is a divergent element. That is, the first two are strengthened by repeated success, encouraging agents to focus in on these successful relational combinations. The third decreasingly effects overall evaluation strength with increasing frequency, encouraging agents to focus on relational combinations infrequently undertaken.

6.1.3. Extension 3: Best-first type approach for working on incomplete design sequences

Finally, the third critical extension is a change extending configuration agents' abilities to allow them to modify and evaluate incomplete design sequences. In the original algorithm, each design is updated until it becomes feasibly complete. These complete designs are the solutions maintained in the solution list from which modifications are made. Each solution in the list is modified sequentially until it becomes feasibly complete, modifying the entire list of designs one step at a time (see Fig. 5). In the new algorithm, on the other hand, agents can evaluate and evolve incomplete solu-

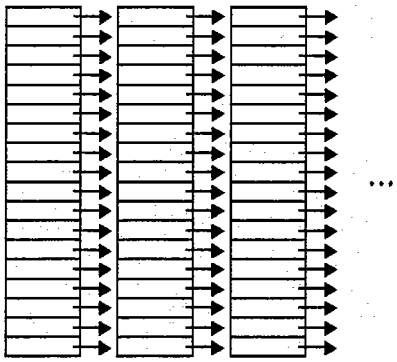

Fig. 5. The solution evolution strategy of the original algorithm, with the entire list of solutions modified one step at a time.

tion sequences as well as completed ones, estimating a projected value based on their relational knowledge. This allows the agents to evaluate and compare the potential impact of modification alternatives as well as the potential final value before the design is complete.

A procedural change was also made to most effectively contextualize the alteration allowing incomplete solution evaluation, and to better match the algorithmic processes to those of human teams. In a best-first like approach (Rich, 1983), the agents of the new algorithm, rather than developing an entire solution list one step at a time, develop a single design at a time. From this solution a list of new solutions is developed. The best of these is taken and likewise developed though various modifications to create again a new list, and thus the process continues. Those promising sequences identified but not developed are evaluated and stored for later consideration (see Fig. 6).

The combination of these two related changes, the evaluation of incomplete design alternatives, and developing individual designs at a time, encompass processes and strategies, from our observations, more closely aligned to those of effective human teams.

6.2. New team-based agent algorithm

This section presents our new team-based algorithm and how the extensions discussed above are implemented. A schematic overview of the new team-based algorithm is presented in Figure 7. This algorithm incorporates three types of agents: configuration and instantiation agents (as with the original algorithm), and a team leader. The team leader acts as a facilitator for the team of configuration agents. Its job is to provide the configuration agent team with starting point designs from which to begin the modification cycle. When the team leader has no design to provide, as is the case for the first cycle, it provides the minimum design sequence: casting → finish machining.

The modification cycle then begins. Each configuration agent examines the given design and generates modification suggestions. The agent suggestions are then aggregated and used to create a set of new, team-modified designs. These designs are then sorted based on the level of agreement between agents' suggestions and projected values as calculated in Eq. (4).

Each of these designs is then evaluated. If a design sequence is not yet complete and feasible it is sent back to the configuration agents, which estimate the design's value collectively. If a design sequence is complete, feasible, and not a repeated design, on the other hand, the design is sent to an internal instantiation loop. In this loop, the design is iteratively instantiated and evaluated using the same probabilistic strategy of the original algorithm. Also, the same final cost functions used in the original algorithm to evaluate designs are also used in this application to assign these completed designs their final objective function values. Once assigned an objective function value, each agent whose corresponding process is represented within the sequence analyzes the design. Each of these agents, subsequently, updates its knowledge according to the process and equations presented in the preceding section. Then, those designs resulting in a sufficiently low objective function value are placed in the final list. This parameter alteration cycle continues until the alterations fail to produce an improved design for a certain number of cycles in a row, which in this implementation is three. All but the top of these designs, both complete and incomplete, are sent to the team leader. The team leader maintains a list of the designs identified by the team, and it is from this list that it draws the designs to present to the configuration agent team as discussed above.

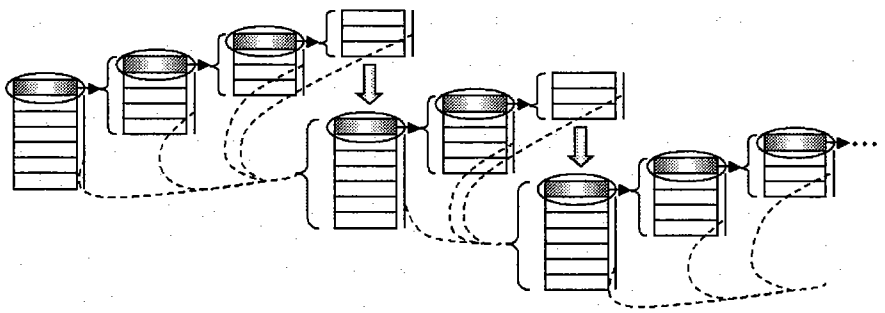

Fig. 6. The solution evaluation strategy of the new algorithm, evolving one solution at a time, that returns to promising solutions identified in the interim following solution development completion.

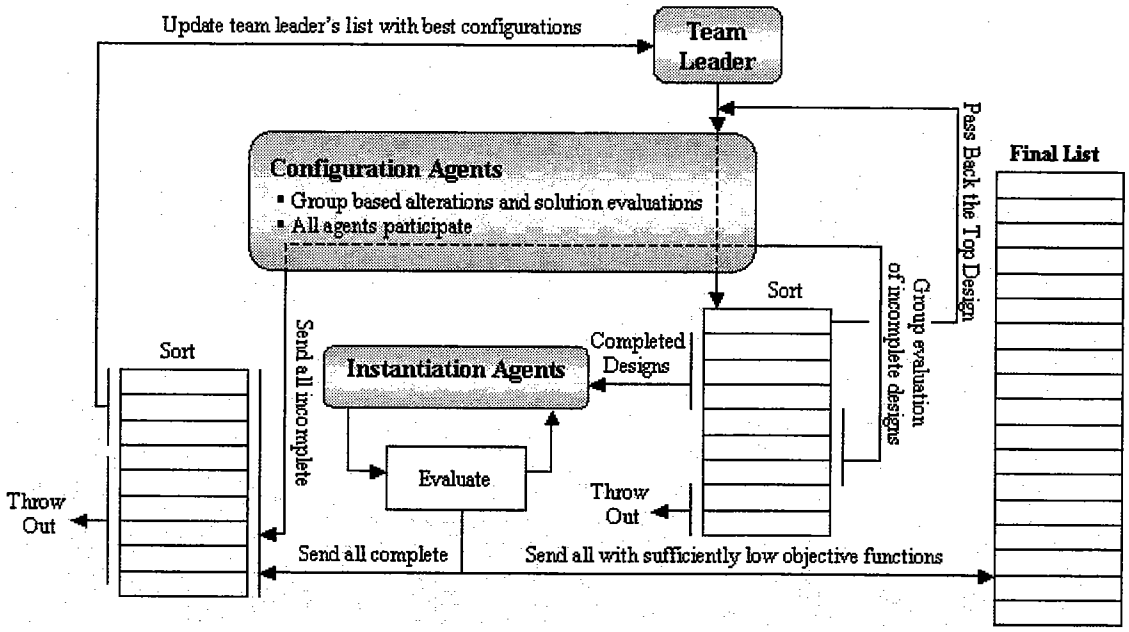

Fig. 7. A schematic overview of the new team-based algorithm.

The top design is recycled back to the configuration agents and the modification cycle is recommenced. Thus, at each iteration the agent team develops the top identified design. When the modifications begin to produce fewer and fewer improved designs, the team leader interrupts the modification cycle and introduces a new design from its list, thereby refocusing the configuration agents' efforts in the most promising new directions.

As the configuration agents begin the modification cycle, as discussed above, they start by individually examining the sequence to determine the "best" modification(s). First, they decide whether to assess process *addition* or process *subtraction* modifications. This decision is made probabilistically, based on the sequence length.

In addition, then, as agents' evaluations are made relative to the location of their respective processes within the sequence, each agent must first determine if and where its process is in the current sequence. If it is not in the sequence, then that agent finds the best location to insert it by evaluating the entire sequence with its process in each potential location, and then chooses the best one. This, then, is one of the agent's modification suggestions; further modification suggestions are then made with respect to that location within the sequence. Next, each agent decides how many (other) changes to make. It does this probabilistically, again based on sequence length, with the maximum number of modification suggestions set at three. For each of these modifications, the agent chooses the insertion location. This decision is also made probabilistically, based on the distance from the agent's location within the sequence, with a preference for modifications closer to the agent. Following this, the agent evaluates all the potential process additions at that location and chooses the best one. The agent continues on for each process addition. Note that each time an agent adds a process it must account for that process in subsequent additions. When completed, each agent has a list of up to three process additions with corresponding projected values and locations within the sequence.

The individual agents then present their suggestions along with their projected values to the team. From this information, group-based modifications are constructed. This is done by listing all possible combinations of the agents' modification suggestions, not in violation of the basis of any of the agent's individual evaluations (i.e., the positional relationship to its process), and ranking them. Those top ranked modification combinations are then applied to the design to produce a set of new designs.

The ranking is based on two factors, group consensus and suggestion strength. Group consensus is based on the similarity between agents' suggestions. This is ascertained through an examination of the suggestion combinations for agreement, in both location and process, between the agents' individual suggestions. An agreement average is taken and the modification combination is assigned a level of agreement (see Fig. 8). Suggestion strength is derived from the projected values assigned to each modification by the suggesting agents [see Eq. (4)]. These projected values are averaged over each modification for the suggestion strength of the total modification combination. If more than one agent agrees on an alteration, the average of their projected values is taken for that modification.

The modification alternatives are then ranked from highest agreement and overall suggestion strength to lowest agreement and strength. As a general rule, the moves with

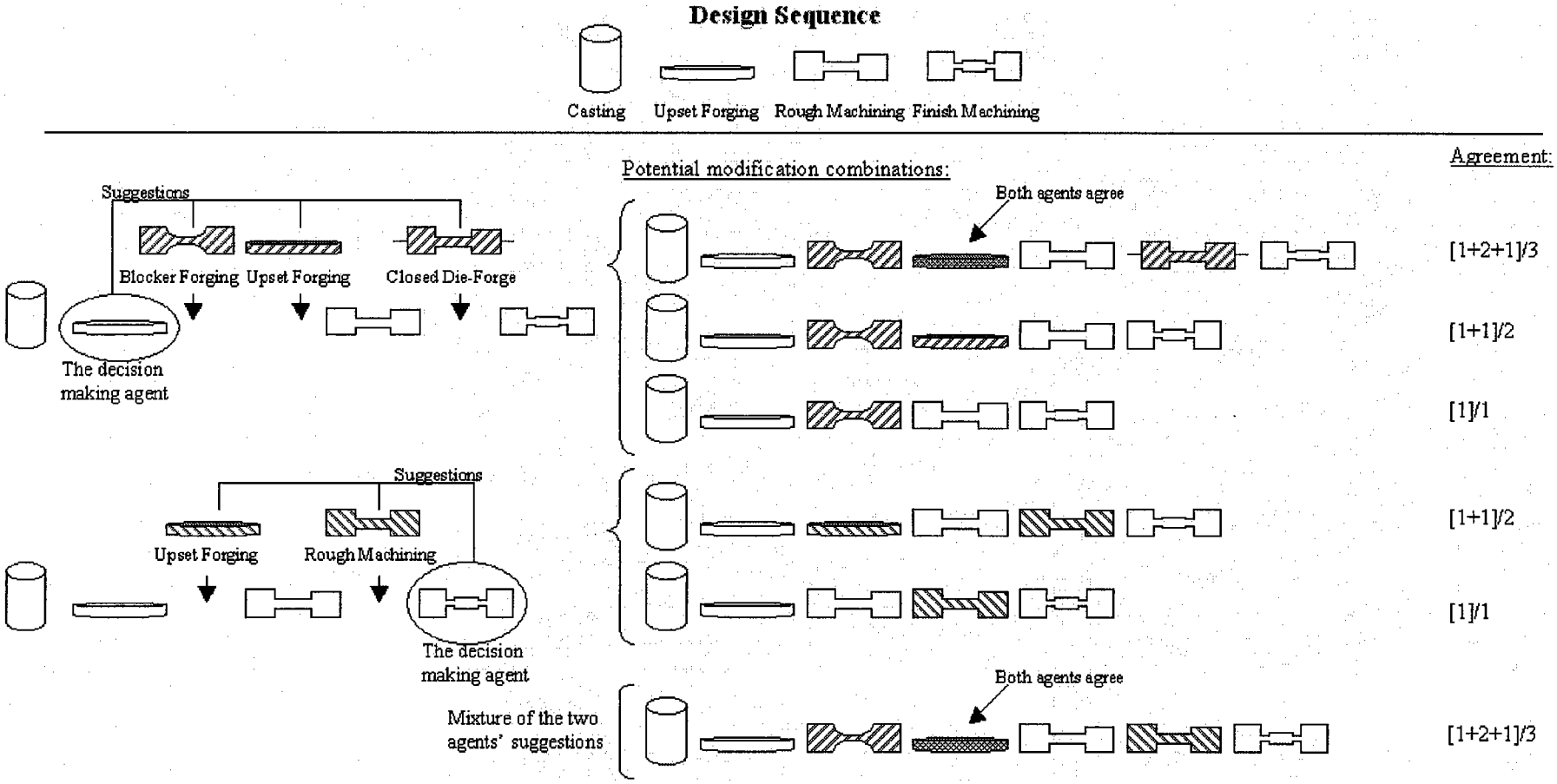

Fig. 8. An example of identifying modification combinations and agreement between the suggestions of two agents.

the most agreement are given the highest priority. This gives priority to the convergence between individuals' views. However, when agreement and suggestion strength are low, and an alternate design change has no agreement but sufficiently high strength, the second is given priority. This is to encourage the acceptance of promising, divergent design changes. The low priority suggestions are then pruned away, and new solutions are formed from the remaining, high-priority suggestions.

If subtraction modifications are considered, each agent, whose process is represented in the sequence, evaluates each other process relative to the location of their process. They then rank these processes from worst to best. Again, when completed, each agent has a ranked list of the processes within the sequence with corresponding estimated value contributions for each. In a fashion similar to that described above, the agents' lists are compared and subtraction combinations are ranked according to agreement and projected value. From this ranking, those processes ranked as the "worst" contributors are subtracted from the design to produce a series of new, modified designs.

Following design modification, as discussed above, those designs that are complete are sent to an instantiation loop, and those designs that are not feasibly complete are evaluated collectively by the configuration agents. The team evaluations of incomplete designs are performed as follows: For each process contained within each incomplete design, the corresponding agent evaluates the processes immediately adjacent, one space to the left and to the right, to its process. Thus, two agents evaluate each process, except the end processes, within each of the design sequences. These duplicate evaluations are then averaged and each design is assigned a total projected value according to

$$\text{projected\_total\_sequence\_value} = \frac{\sum(\text{averaged\_process\_evaluations}) + \text{end\_process\_evaluations}}{\text{sequence\_size}}. \quad (5)$$

For an overview of the entire team-based algorithm see Figure 9.

7. RESULTS: ALGORITHM PERFORMANCE

7.1. Comparison of results

Both programs were run 30 times, each at 20,000 iterations per run. During each run, the programs output a running count of the completed designs, their objective functions, and their time of completion. These data were averaged and analyzed over the set of runs for each of the programs.

The new team-based algorithm took substantially more time to run to completion than the original separable algorithm. On average it took 2.7 times longer per iteration than the original. However, the new agent method proved to be much more efficient in identifying good (physically and economically feasible) solutions. For example, while the original algorithm identified on average just 1.2 good designs in a thousand iterations, the new team-based algorithm identified 123 per one thousand iterations. So, in the same amount of time, the new method located 894 good designs, whereas the original method identified just 23.4 (refer to Fig. 10). Identification of good designs per unit time increased an entire order of magnitude, a significant improvement.

Both algorithms regularly converged on the same top sequence: casting → upset forging → upset forging → rough machining → finish machining. Nevertheless, the quality of designs, both average and best, identified by the new team-based algorithm, saw a significant improvement over the original separable algorithm. The best solutions found in the same time period, averaged over the 30 runs each, had objective function values of 9974.5 and 7194.1 for the original and the new agent algorithms, respectively. This represents an improvement of nearly 140%.[1] Figure 11 depicts the best objective functions identified over time for the two algorithms, averaged over the 30 test runs each. Likewise, the average objective function in the final list, again in the same time period, for the original and new agent algorithms were 20760.1 and 7329.9 respectively, an improvement of more than 280%. Figure 12 depicts the average objective function in the final list over time for both algorithms, averaged over the 30 test runs each. Further scrutiny shows that on average nearly every design in the new agent method's list was better than even the best located by the original.

It is interesting to note that the new team-based algorithm was also generally more successful at improving the same sequence configurations through parameter alterations. For example, in the original algorithm the very best configuration solution in all 30 runs had an objective function value of 7189.1. The new algorithm, alternatively, completed the same configuration with a minimum objective function value of 7139.1. See Table 2 for the top design configurations and their associated minimum objective function values.

7.2. Statistical evaluation: Factorial experiment

The level of improvement over the original algorithm was significant and surprising. The new team-based algorithm was a great deal more effective in traversing the solution space, identifying higher quality and a much broader range of design solutions. To identify the sources of differentiation and their degree of impact, we created and ran a factorial design of experiments. Specifically, we were inter-

[1]Note that the results for the original method in this study differ somewhat from those published in Deshpande and Cagan (2003). The best solutions found averaged 7223.2 in that publication. This discrepancy is due to the algorithm alterations prohibiting repeated designs.

```
Randomly generate initial population (both processes and parameters), set up processes, & evaluate designs
while (# of iterations < max. iterations)
    Team leader chooses top undeveloped sequence in his list for development
        if (Team leader's list is empty)
            Choose minimum sequence to start: Casting→ Finish Machining
    limit = 0
    while (progressing < constant_a and limit < constant_b)
        if limit > 0
            Start with top sequence identified in previous round
        Agents propose configuration modifications
            Probabilistically choose to add or delete processes based on sequence length
            if adding, for every agent:
                Check to see if "agent's" process is in sequence
                    if no:
                        Locate the best position for it
                    if yes:
                        Randomly choose one (if it is present more than once)
                alt = total number of modifications chosen probabilistically
                while (alterations < alt)
                    Probabilistically choose location for current alteration
                    Evaluate the best process for that location based on relational knowledge
                    Estimate values of quality, confidence, and frequency
                Make suggestions for process additions with their respective values
            if deleting:
                Agents responsible for each process in the given sequence evaluate the values of each process
                They, then, each make suggestions on which processes should be deleted
        Check for agreement
        Rank alterations according to agreement and estimated suggestion values
        Save top design sequence for next round
        Generate new design sequences from those with agreement and/or sufficiently high estimated values
        Check each new design sequence
            if feasibly complete:
                if it is not a repeated design
                    Probabilistically assign parameters
                    Evaluate
                    Update memory
                    if sufficiently low objective function, replace lowest rank design in list, maintaining a sorted list
                    stop = 0
                    while (stop < constant_c)
                        Generate a new design by probabilistically modifying the parameters
                        Evaluate
                        Update learning
                        if the objective function is higher than the previous design's
                            stop = stop + 1
                        else
                            stop = 0
            if not feasibly completed:
                Group evaluation and assignment of projected value
        Sort new design sequences according to their values / projected values
        Store designs with sufficiently high evaluations in team leaders list
        if none of the new designs resulted in an estimated value greater than the initial sequence
            progressing = progressing + 1
        limit = limit + 1
```

Fig. 9. The high-level pseudocode overview of the new team-based algorithm.

ested in assessing the effects of five key elements on the performance of the new algorithm. Each variable was evaluated at two levels, present or not present. The test variables were interagent ties, domain knowledge, and relational knowledge.

7.2.1. Interagent ties

This was a lumped variable that included all of the collaborative aspects of the new team-based algorithm. It incorporated the sharing of solution suggestions between design