An Agent Based Optimization Approach to Manufacturing Process Planning

Saurabh Deshpande

Jonathan Cagan
e-mail: cagan@cmu.edu

Computational Design Lab,
Mechanical Engineering Department,
Carnegie Mellon University,
Pittsburgh, PA 15213

Many optimization problems, such as manufacturing process planning optimization, are difficult problems due to the large number of potential configurations (process sequences) and associated (process) parameters. In addition, the search space is highly discontinuous and multi-modal. This paper introduces an agent based optimization algorithm that combines stochastic optimization techniques with knowledge based search. The motivation is that such a merging takes advantage of the benefits of stochastic optimization and accelerates the search process using domain knowledge. The result of applying this algorithm to computerized manufacturing process models is presented. [DOI: 10.1115/1.1641186]

1 Introduction

Many optimization problems result in models of quite complex spaces. Consider the design of bulk manufacturing process plans for industrial forging processes (Fig. 1). Sequences of hot and cold forging, heat treating (not shown), and machining lead to a final configuration; these sequences may include repeated process selection or return to previously used processes. Within each process, parameter values must be selected as well. The optimization problem becomes one of optimal process selection in conjunction with optimal parameter selection, and the optimization space is quite complex—discontinuous and highly non-linear/multi-modal.

Due to the nature of the space of this and other design configuration problems, researchers have moved toward stochastic and heuristic optimization methods with the goal of obtaining a very good, rather than guaranteed optimal solution.

Success of simulated annealing and genetic algorithms, for example, has demonstrated promise of these types of approaches. However the optimization space is vast, and the reliance on near randomness alone makes the task of finding an acceptable solution difficult. Consider the continuum shown in Fig. 2 between deterministic and random optimizing search strategies (adapted from Yin [1]).

Although gradient-based strategies are efficient, they have difficulty finding an acceptable optimum; genetic algorithms and simulated annealing, on the other hand, are effective in seeking out acceptable optima, yet often require an unacceptable amount of time to do so. As shown in the figure, we propose that an algorithm better able to use deterministic information within a stochastic framework has the potential to most efficiently solve such problems.

This work introduces a new approach to optimal design. The proposed approach is to use agents within a stochastic framework to configure and modify designs. The agents are a modification over the move sets typically used in simulated annealing. The move sets change one parameter or set of parameters randomly. Agents not only have the functionality of move sets but also have domain knowledge to probabilistically modify a given design state towards an optimum state—the knowledge is deterministic, but its application is done strategically and probabilistically keeping within the approach a stochastic/random flavor. Designs are pooled into a population; the creation of a population is considered an iteration or generation. At each iteration, designs are evaluated and ranked. Inferior designs are pruned away while remaining designs form a basis from which the remaining population is formed. This population aspect of the approach resembles iteration within genetic algorithms. However the method differs from traditional genetic algorithms in that modification of the designs range from stochastic to more deterministic changes but are explicitly goal-directed, and the creation of designs is heavily influenced from deterministic knowledge about the problem domain.

The work presented in this paper also extends the basic Agent based method to solve the configuration problem. The method here is implemented by maintaining a population of designs that have not only different process parameters but also different process sequences. The design modification here is done by two types of agents: sequence agents and parameter agents.

Sequence agents modify the process sequence by adding or deleting processes. By using a simple constraint satisfaction method they ensure that all design sequences are feasible, while parameter agents modify the parameters associated with processes as above by probabilistically using domain knowledge specific to manufacturing.

The next section presents the necessary background. Section 3 discusses and presents ideas for creating agents for bulk manufacturing process optimization. Section 4 presents the "base parameter optimization algorithm" using agents and the results associated with it. Section 5 presents various modifications to the base algorithm and demonstrates improvement over the base algorithm for each modification. After presenting and justifying each modification individually, in Section 6 all modifications are combined into one integrated algorithm called the "agent based parameter optimization algorithm" with superior performance to the base algorithm and each individual modification. Section 7 applies the agent based parameter optimization algorithm to the continuous problem where parameters are allowed to take on any real number within a defined range. Section 8 extends the method to sequence modification, called Integrated Agent based Configuration Algorithm, and Sec. 9 presents a description of constraint satisfaction for ensuring feasible sequences. Section 10, presents and discusses the results of applying the configuration algorithm to the turbine disk manufacturing process. Finally, Sec. 11 concludes with insights on the work.

2 Background

2.1 Problem Definition: Bulk Manufacturing Process Planning. Manufacturing process planning involves deciding on a plan to manufacture a part based on minimum cost criteria. The plan outlines the selection of the manufacturing processes (e.g. casting, blocker forging, rough machining etc.), sequencing of the processes and the parameters to be used for each manufacturing process.

Contributed by the Design Theory and Methodology Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received July 2001; rev. June 2003. Associate Editor: D. L. Thurston.

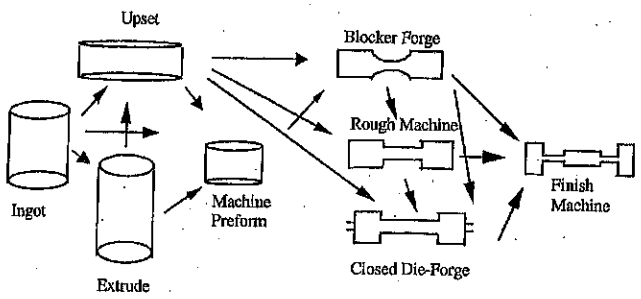

Fig. 1 Forging process schematic

Specifically, the input to a manufacturing process planning problem is a description of the part to be manufactured in terms of geometry and material data. The output is the set of processes with the associated process parameters that can be used to manufacture the part with the least possible cost. The optimization problem is to find the optimal process sequence with the associated optimal process parameters for manufacturing the part with minimum cost.

Formally,

Let S be the optimum sequence with the associated optimum parameters.

Let S_i be a sequence consisting of "n" processes P_{i1}, P_{i2}, $P_{i3} \ldots P_{in}$, and let $S_{i,\min}$ be the optimum objective function value of sequence S_i with the associated optimum process parameters.

Then,

Find S such that $S = \min(S_{1,\min}, S_{2,\min}, S_{3,\min} \ldots S_{i,\min} \ldots S_{m,\min})$;

$$S_{i,\min} = \text{Min Cost}(S_i) = \sum_{j=1}^{n} \text{Cost}(P_{ij}) + \text{Miscellaneous Costs}$$

where j = no. of processes in a sequence and

$\text{Cost}(P_{ij}) = f(P_{ij1}, P_{ij2}, P_{ij3} \ldots P_{ijk})$;

All costs are in U.S. dollars.

P_{ij1}, P_{ij2}, $P_{ij3} \ldots P_{ijk}$ are k design parameters associated with process P_{ij}.

For the parameter optimization problem with a fixed sequence $i = 1$.

The parameters associated with the processes used are as follows:

1. Casting: Billet Radius, Billet Height
2. Upset and Blocker forging: Die geometry, Die temperature and Die Friction Factor
3. Preform, Rough and Finish Machining: Geometry.
4. Extrusion: Die geometry, Die Friction Factor, Die length, Die temperature and Die velocity

Additionally, penalty terms are introduced into the calculation of process costs. For forging operations a penalty term is introduced when the press capacity is exceeded and aspect ratio (height to radius ratio) is larger than a specified limit (in this case 3). For machining, a penalty term is included when geometry of the final workpiece does not match the desired workpiece after machining. Also, during final microstructure evaluation a penalty term is introduced when the final microstructure of the workpiece violates the specifications. The miscellaneous costs include heat treatment and ultrasonic nondestructive evaluation costs. Work done at Wright Patterson Air Force Base and at Ohio University has dealt with calculation of the cost of individual processes (Fischer et al. [2], Gunasekera et al. [3]). Previous implementation of the cost function was limited to finding the cost of a small number of process sequences. In this work the cost function calculation is modified such that it can find the cost of a large number of process sequences including those in which processes are rearranged and processes are present multiple times.

Gradient Based Optimization	***Agent Based Optimization***	Simulated Annealing	Genetic Algorithms
Deterministic			**Random**

Fig. 2 Continuum of optimizing search strategies

2.2 Previous Approaches. There are two main approaches to developing process plans: variant and generative (see Chang et al. [4]). Our work falls under the category of generative process planning optimization. In it the features are parameterized making it amenable to optimization techniques. Previous work using a complementary approach by Jacobson et al. [5] assembled together a class of heuristic optimization methods under a common framework called Generalized Hill Climbing (G.H.C). Specifically, they have investigated five types of G.H.C's namely Simulated Annealing, Threshold Accepting, Monte Carlo Search, Local Search and Weibull Accepting with three neighborhood rules. Their results provide a point of comparison for the method presented in this paper. A recent modification (Vaughan et al. [6]) has involved application to variable process sequences using a new neighborhood function and a switch function, which provide a framework to move between different process sequences. However, the work is limited in scope as only a limited number of sequences were explored (5 sequences) and also because the method does not take into account multiple processes of the same type.

Process planning using agents on NC turning centers has been investigated by Storr et al. [7]. Our work differs from Storr et al. [7], in that we consider a variety of manufacturing processes (casting, forging and machining) and seek out the optimal process plan; in particular, our emphasis is on the introduction of an optimizing agent-based algorithm with general application.

2.3 Stochastic Algorithms. In this section we present a brief description of three stochastic algorithms: Genetic algorithms, simulated annealing and A-Design.

Genetic algorithms were introduced by Holland [8] as a computational model of biological evolution. The aspect of genetic algorithms that is used in agent based optimization algorithm is maintaining a population of designs. Having a population of designs has the advantage of parallel exploration of the search space.

Another method from which some ideas have been borrowed is simulated annealing (Kirkpatrick et al. [9]). The driving idea behind simulated annealing is that initially poor designs are all accepted (leading to exploration of the search space) and towards the end poor designs are only occasionally accepted (leading to exploitation of the search space). A similar approach is used in the agent based optimization algorithm introduced in this paper where initially large changes are made to the design state and towards the end only smaller changes are made. In more advanced simulated annealing algorithms, move steps are taken probabilistically based on how the algorithm performs during execution. This method, called the Hustin move set (Hustin et al. [10]), is also extended and used within the present work.

The current research also finds its roots in the A-Design method of Campbell [11] and Campbell et al. [12]. The approach taken in A-design is to use design agents to act on a representation of the problem domain to create design configurations, instantiations and modifications. Similar to A-Design, a collection of agents is used in our work to modify the design state. We differentiate the agent based optimization algorithm from A-Design in terms of 1) the problem representation, 2) the types of agents utilized, 3) the level of knowledge used within the agents (A-Design used very little domain-specific knowledge and still worked toward the extreme of randomness while the current work uses more specific knowledge about the domain).

3 Optimizing Agents

As defined in Campbell et al. [12], and based on the ideas from the A-Life community (Langton [13]) and the definition from

Russell et al. [14], agents are viewed as perceiving their environment (here, the design state), making judgments of how to effect change on a design state, and then acting upon their environment through effectors (i.e., modifications to the design state). In our model agents collaborate with each other by taking each other's solutions and modifying them. By having agents with different abilities and preferences collaborate the process gains robustness and variety in solving different problems and leads to emergent behavior.

Various types of agents have been explored in the literature in the area of process planning and engineering design. Storr et al. [7] present an agent based process planning framework for machining on NC machines. Mori et al. [15] have used agents for collaborative design. Our approach is unique in that it combines process planning with optimization in an agent framework.

In this paper domain knowledge is incorporated but agents are used in a stochastic framework. We restrict our focus to design agents involved in optimization. For creating agents we develop an analogy with human group activity. In real world problems teams of engineers or designers work on a problem with each member of a team having expertise in a different area. Similarly, for design optimization purposes, we first identify the different aspects of design and make agents for each of them. Agents have domain knowledge about the specific aspect of design considered and use it to change the parameters associated with design. They also keep statistics of how successful they have been in the past and based on that make a decision about whether to change a design parameter.

3.1 Bulk Manufacturing Parameter Optimizing Agents. Consider the bulk manufacturing process planning problem which involves a collection of manufacturing processes. Agents are made corresponding to each manufacturing process that not only change associated parameters but also have domain knowledge about the process. Fifteen instantiation agents specific to the bulk manufacturing domain have been created. These agents are focused around the geometry of the workpiece and process properties. As discussed before, these agents address the problem of optimizing parameters given a process sequence. In any operation the goal is to most efficiently reach the desired geometry without removing more material than allowed.

These instantiation agents are as follows:

a. For most processes (casting, upset forging, machine preform, blocker forging, closed die forging and rough machining) we have one geometry agent each for deciding the dimensions of the workpiece.
b. In addition, for upset forging, closed die forging, and blocker forging we have three agents each, as listed below:

i. An agent for changing the die speed.
ii. An agent for changing the die temperature.
iii. An agent for changing the die friction factor.

These agents decide what parameter values to select or to modify. The goal is to enable the agents to select sub-optimal local choices under the supposition that compensating for those inferior choices may better be made through other processes later. Thus there is a preference to being efficient about selecting the best parameter values (and thus the use of domain knowledge in the agents), but to only apply that knowledge probabilistically to enable for alternative solutions to emerge.

An example is the casting agent that has domain knowledge specific to bulk manufacturing. In bulk manufacturing, the initial workpiece volume should always be more than the final desired workpiece volume as otherwise it is not possible to reach the final workpiece geometry through any of the bulk manufacturing operations (casting, upset forging, open die forging, blocker forging, closed die forging, rough machining and finish machining). On the other hand, with a very large volume, although the workpiece can be operated upon to generate the final workpiece, the machining operations become costlier as excess material has to be removed from the workpiece to manufacture the final workpiece. This leads us to a heuristic that the casting agent should select the initial workpiece geometry such that the workpiece volume is within a lower and upper bound. Another example of domain knowledge within the casting agent is regarding the aspect ratio. The casting agent will only generate geometries where the aspect ratio (height/radius ratio) of the workpiece is less than a specified limit.

1. Enter the population size and the number of generations (gen).
2. A priori select the appropriate instantiation agents associated with the process sequence.
3. Decide the probability of selection of the instantiation agents (prorata basis).
4. Generate initial population.
5. I= 1
6. While (I <= gen)
   - Sort the population in increasing order of cost;
   - Categorize the population into Good (top 50%) and Poor (remaining 50%);
   - Reject Poor;
   - Generate new members such that each member of Good population generates one new member. This is done by probabilistically selecting an instantiation agent that modifies the parent member from Good population to generate a new member;
   - I=I+1.

Fig. 3 Base parameter optimization algorithm using agents for a given sequence of processes

4 Base Parameter Optimization Algorithm

The base parameter optimization algorithm uses ideas from genetic algorithms and the A-Design method as appropriate towards a foundation of optimizing search balanced between random and deterministic search.

The base algorithm for a given sequence of processes can be summarized as shown in Fig. 3. Given a sequence, the values of the parameters must be determined via instantiation agents. As already mentioned, these agents may have general guidelines on how to make parameter changes based on how the processes are represented, or they may have domain or process specific knowledge. The goal of the process specific knowledge is to accelerate the optimization process—these agents ensure a more intelligent search of the design space leading to faster convergence to optima. The knowledge is used as a guideline to the system to *probabilistically* make changes to a design—there is *always* a chance for a stochastic change rather than a pre-determined modification to the design state. This enables the algorithm to explore different regions of the search space and avoid inferior local optima.

The iterative strategy is similar to A-Design in that a population of designs is maintained. In each iteration the designs are evaluated and sorted based on their objective function values. The bottom half of the population is discarded and in its place new designs are created by modifying copies of the remaining members of the population. The idea is that, as with genetic algorithms, those better designs provide a basis from which even better designs might evolve. The end result is that the population as a whole progresses toward better and better solutions. The process continues until convergence is reached or the allocated number of iterations (or generations) is completed.

4.1 Results. The algorithm has been coded in GNU C and tested on a Silicon Graphics Octane workstation with 192 MB RAM and 195 MHz, MIPS R10000 CPU. It has been run on three sequences found in Jacobson et al. [5] for 30 different random number seeds. For each of the three sequences a population of size 100 was kept and the algorithm was run for 100 generations (i.e., 50 new designs were generated at each generation). The total

Table 1 Comparison of GHC algorithm and base algorithm

	Seq.	Min.	Max.	Std. Dev	Avg.
GHC Alg. (10000 evaluations)	1	1919.3	1927.0	2.2	1921.7
	2	2238.4	2244.2	1.1	2238.6
	3	2245.5	2282.7	19.1	2264.3
Base Parameter Optimization Alg. (5000 evaluations)	1	1919.3	1919.3	0.0	1919.3
	2	2238.4	2238.4	0.0	2238.4
	3	2245.5	2245.5	0.0	2245.5

number of objective function evaluations was 5000 (50 designs $\times$ 100 generations).[1] In each generation, after sorting the population, the bottom 50% (in this case 50 designs) were rejected, and correspondingly 50 new designs were generated. These new designs were generated by probabilistically selecting an appropriate agent, which then modifies the associated parameters using domain knowledge or randomly.

Although the probability of selecting an agent, a parameter, or the amount of change will be done probabilistically by examining dynamic statistics maintained on the performance of the algorithm in the next section, at this time the only level of probabilistic selection that has been implemented is the probability of selecting an agent. In this base algorithm the probabilities are static and are allocated on a prorata basis with each agent's probability of selection being directly proportional to the number of parameters it changes. To illustrate, if there are 2 agents (A and B) with agent A changing "n" parameters and agent B changing "m" parameters then the probability of selection of agent A is $n/n+m$ and of agent B is $m/n+m$.

Evaluation is an important part of the design process. The evaluation is done by process models supplied by Wright Patterson Air Force Base (WPAFB) (Fischer et al. [2], Gunasekara et al. [3]) for each of the manufacturing processes; in the future alternative models can be used within our optimization framework. At each iteration these process models serve to simulate the performance and thus cost of a given manufacturing sequence. The objective of the optimization algorithm is to find the manufacturing process design with the least possible cost.

At this time we have discretized the parameter values as was done in Jacobson et al. [5]. In Sec. 7 we will consider the continuous problem with the expectation that the further refinement will lead to improved performance. However the current discretization enables a more direct comparison with current results in the literature.

Table 1 shows a comparison of results from our base algorithm and the best GHC algorithms found in Jacobson et al. [5]. The algorithm has been implemented on three process sequences, which will be referred to as Sequence 1, 2 and 3 in the paper:

a. Sequence 1: Casting→Upset Forging→Rough Machining →Finish Shape Machining
b. Sequence 2: Casting→Upset Forging→Blocker Forging →Finish Shape Machining
c. Sequence 3: Casting→Upset Forging→Machine Preform →Blocker Forging→Finish Shape Machining

The best GHC algorithm was found to be different for each of the three sequences:

a. Sequence 1: Threshold acceptance with neighborhood rule 2;
b. Sequence 2: Weibull acceptance with neighborhood rule 2;
c. Sequence 3: Simulated annealing with neighborhood rule 2.

[1]In actuality the exact number is 5100 (5000 designs+100 initial designs) instead of 5000 as there are 100 initial designs. For presentation of the results, the evaluations are rounded to the nearest multiple of 500, so for 30, 50, and 100 generations there are 1500, 2500, 5000 evaluations instead of 1600, 2600, and 5100 evaluations respectively after rounding.

For comparison we use the same evaluation routines from WPAFB enabling a fair comparison to the results in the Jacobson et al. [5] paper.[2] However, it is important to note that from this comparison we do not intend to draw any conclusions regarding the quality of one approach versus the other. It is clear that the GHC simulated annealing algorithm, for example, could be improved with a dynamic annealing schedule and, as discussed in the next section, many further modifications to our base algorithm will improve the results over those presented in Table 1. However, the comparison is useful in that a favorable comparison indicates the approach worthy of further exploration.

As can be seen from Table 1, our base algorithm using agents consistently finds the optimal solution for all random number seeds for all the process sequences. In each case the overall performance of our algorithm was better than the best GHC algorithms for each problem and required half the number of objective function evaluations. Further, with our approach only a single algorithm needed to be run on each problem to obtain the best solution. These sets of experiments have shown that our base algorithm using agents is comparable to GHC algorithms and gives better results for the three process sequences.

5 Modifications

In the previous section the base parameter optimization algorithm was outlined. To improve the performance of the base algorithm various modifications are presented in this section. A detailed description along with the results of applying each of the modifications individually follows.

5.1 Modification 1: Dynamically Adjust Probabilities of Selecting Each Agent by Increasing the Probability of Agents That Have Been Successful in the Previous Generation. In the base algorithm the probabilities of selection of the agents is kept fixed throughout the iteration. Because of this, even if an agent is unsuccessful, its probability of selection remains unchanged and it is called the same number of times as before. The rationale for modifying the probabilities dynamically is that it will take into account an agent's success history, with higher probability of selection for an agent if it has been more successful relative to others.

We introduce a technique to dynamically adjust the probabilities of selecting agents by a procedure based on the Hustin move set (Hustin et al. [10]) found as a modification to simulated annealing. In simulated annealing, the probabilities of applying various moves are updated after each temperature reduction based on the performance of those moves at the previous temperature. In our case, an analogy is formed between "temperature" and "generation," so that the probabilities of selection are modified after each "generation." In addition, we form an analogy between move set and agents, so that the probabilities of selection of the move set correspond to the probabilities of selection of agents. As a measure of performance, for each of the agents a quality factor is calculated. The quality factor for the i^{th} agent is defined as:

$$Q_i = \frac{1}{n_i} \sum_{\text{acceptedsteps}} |\Delta C|, \quad i = 1, \ldots m,$$

where

n_i = number of times the i^{th} agent was called at the previous generation,
ΔC = change in cost due to an accepted step; an accepted step is one which reduces the objective function value,
m = number of agents.

If an agent is not called in a particular generation then its quality factor is set equal to zero. Also, to ensure that each agent could potentially be called, each agent is assigned an initial probability

[2]Thanks to the generous interaction with Prof. Jacobson we were able to run the GHC algorithms with our version of the WPAFB evaluator to confirm consistent evaluations with the results from their paper.

of selection. This initial probability could be uniform for all the agents or there could be an initial bias if there is some preexisting domain knowledge. However the sum of all initial probabilities is 1:

$$\sum P_{\text{initial proportion},i}=1,$$

where $\Sigma P_{\text{initial proportion},i}$ is the initial probability of selection of all the agents.

For the case when each agent is assigned the same initial probability of selection, $P_{\text{initial proportion}}$ is the same for all agents, and when there is bias then $P_{\text{initial proportion}}$ is different for some agents. In this application, the initial probability of agents is directly proportional to the number of parameters it changes (higher if it changes more parameters) and is kept fixed throughout the iteration.

We define one more factor called P_u (with $0<P_u<1$), which ensures a relative contribution from the fixed initial bias and trades this off with the dynamic calculation of the quality factors (Q_i). The probability of selection of the i^{th} agent is then given by:

$$P_i=P_{\text{initial proportion},i}*P_u+(1-P_u)\frac{Q_i}{Q_{\text{total}}},\quad i=1,\ldots m,$$

where Q_{total} is the sum of the quality factors for all the agents. As can be inferred, the larger the quality factor for an agent the greater the probability that it will be applied at the next temperature.

5.2 Modification 2: Adjust the Amount of Perturbation to Each Design During Modification by Increasing the Perturbation as the Algorithm Modifies Worse Performing Designs. In this modification, the selected population is sorted in increasing order of cost and then, based on a general curve, in this case linear, a variable number of agents are called so that fewer agents are called for population members that have lower objective function values and more agents are called for population members that have higher objective function values. This promotes small changes to the process sequence parameters whose members have lower objective function values (as they are closer to optimal) and larger changes to members that have higher objective function values (as they are further away from optimal) thus increasing the chance that they will reach the optimum faster. In this application, the agents have a prorata probability of selection that is fixed initially.

The formula for the curve is:

$$\text{Number\_of\_agents}=\frac{\text{pop\_counter}}{K}+1,$$

where

pop_counter = place in sorted population in a given generation,
K = constant, which is fixed in each process sequence before start of simulation.

5.3 Modification 3: Use the Concept of the Annealing Temperature from Simulated Annealing to Dynamically Adjust the Number of Small Versus Large Perturbations Across the Population Pool as the Algorithm Runs to Encourage Convergence. In simulated annealing, based on the annealing schedule poor designs are accepted initially to explore the design space and then, as the iterations proceed, the focus shifts toward only accepting progressively better designs. We adopt a similar philosophy whereby large changes are initially made to the design process parameters (by calling more number of agents) and progressively smaller changes are made as the iteration proceeds.

The formula used to determine how many agents to call is:

$$\text{Number\_of\_agents}=\frac{\text{pop\_counter}}{(\Delta+\text{gen\_counter})}+1,$$

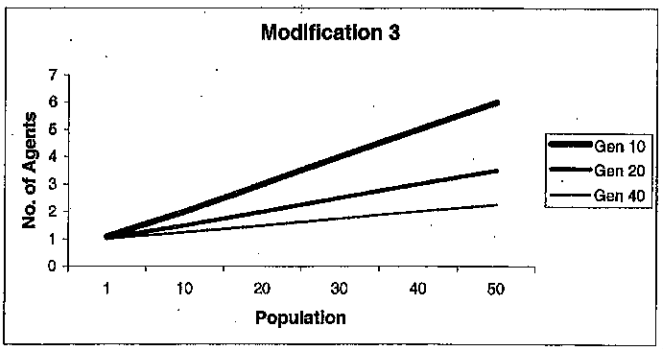

Fig. 4 Decreasing number of agents called as algorithm progresses

where:

pop_counter = place in sorted population in a given generation,
gen_counter = generation counter,
Δ = constant, which is fixed in each process sequence before start of simulation.

As desired, at a given generation (gen_counter=constant) the number of agents called increases with "pop_counter," i.e. for a sorted population more agents are called for the inferior designs in accordance with Modification 2. Also, the number of agents called decreases as gen_counter (generation counter) increases, in accordance with Modification 3. This is depicted graphically in Fig. 4, where three curves are plotted for Generation 10, Generation 20 and Generation 40.

5.4 Modification 4: Application of Positive Feedback. As presented in Section 5.1 (Modification 1), the Hustin move set is an effective way of providing positive feedback to agents. However, its effectiveness is limited as it takes into account the performance of only the previous generation in the iteration process. If we also take into account the agent's success history over all the previous generations and use that information to modify the agent probabilities then it will increase the chances of reaching the optimum faster. This is based on the assumption that agents which have been successful in the past will be successful in the future, and also because using an agent's success statistics throughout all the previous generations gives a larger pool from which to select the agent probabilities rather than just the previous generation. Hence, as the iteration proceeds, the performance characteristics of each agent over all generations is taken into account and not just the previous generation (as in Modification 1).

Let P_i=Probability of selection of i^{th} agent and $P_{\text{initial proportion},i}$=initial prorata probability of selection of i^{th} agent.

Given:

$$P_{s,i}=\frac{\text{No. of successes of } i^{\text{th}} \text{ agent throughout the generations}}{\text{Total number of times the agent has been applied}},$$

$P_{s,\text{total}}=\Sigma_{i=1}^{n}P_{s,i}$, where n=total number of agents,
P_u=Proportion of prorata fixed probability (determines the trade-off of prorata probability with positive feedback),
then

$$P_i=P_{\text{initial proportion},i}*P_u+(1-P_u)*\frac{P_{s,i}}{P_{s,\text{total}}}.$$

As shown above P_i is formed from two terms: $P_{\text{initial proportion},i}$ and $P_{s,i}/P_{s,\text{total}}$ which comes from the proportion of the number of times the agent has been successful in the previous generations, with P_u ($0<P_u<1$) deciding the relative weight of each.

5.5 Modification 5: Application of Negative Feedback. In this modification negative feedback is applied to agents by looking at their failure statistics and reducing the probabilities of agents that have been unsuccessful. By dynamically modifying the probabilities of agents in this way, the chance that an agent which has been unsuccessful in the past is selected is reduced, thus increasing faster convergence to optima by reducing wasteful search. This modification is based on the assumption that an agent that has been unsuccessful in the previous generations will have a higher likelihood of being unsuccessful in future generations as well. This is in contrast to Modification 1 (Hustin move set) and Modification 4 (positive feedback history), both of which used positive feedback, wherein agents which have performed better are rewarded by increasing their probability of selection.

Let P_i=Probability of selection of i^{th} agent and $P_{\text{initial proportion},i}$=initial prorata probability of selection of i^{th} agent.
Given:

$$P_{f,i}=\frac{\text{No. of failures of } i^{\text{th}} \text{ agent throughout the generations}}{\text{Total number of times the agent has been applied}},$$

$P_{f,\text{total}}=\Sigma_{i=1}^{n}P_{f,i}$, where n=total number of agents,

P_u=Proportion of prorata fixed probability, determining the trade-off of prorata probability with negative feedback,
then

$$P_i=P_{\text{initial proportion},i}*P_u-(P_u-1)*\frac{P_{f,i}}{P_{f,\text{total}}}.$$

As shown above P_i is formed from two terms: $P_{\text{initial proportion},i}$ and $P_{f,i}/P_{f,\text{total}}$ which comes from the proportion of the number of times the agent has been unsuccessful in the previous generations, with P_u ($1<P_u<2$ as there is a negative weight given to the negative feedback term $P_{f,i}/P_{f,\text{total}}$ for reducing the probability) deciding the relative weight of each.

5.6 Results of All Modifications. All the simulations were performed using 30 different random number seeds. The population size was kept constant at 100. Simulations were performed until 2500 evaluations (50 generations×50 new members at each generation ≅2500 evaluations). For sequence 1 and 2 the results obtained could not be distinguished as the algorithms converged fully at 2500 evaluations, thus for comparison purposes the results at previous generations (1500 evaluations) are shown instead.

The results of all the modifications applied individually are presented in Table 2 and compared with the base algorithm. For sequence 1 and sequence 3, observe that the algorithms with individual modifications (Modification 1 to 5) have lower maximum, standard deviation and average compared to the base algorithm. For sequence 2, the algorithms with individual modifications (Modification 1 to 5) have lower average and standard deviation compared to the base algorithm.

6 Agent Based Parameter Optimization Algorithm

As the results in the previous sections show, each of the individual modifications has improved the performance of the algorithm over the base algorithm. This suggests that all the modifications have properties desirable in an optimization algorithm and so an approach combining all the modifications into one algorithm is promising. This section investigates combining all the modifications into one algorithm for parameter optimization and compares its performance with the previous results.

6.1 Modification of Hustin Move Set for a Multi-Agent Scenario. The Hustin move set was designed to reinforce those moves that individually improved the objective. In a multi-agent scenario when there are many agents modifying a design there is a problem of credit assignment. This issue becomes crucial as only those agents that have reduced the objective function value should be rewarded. Other agents that have not contributed to reducing the objective function value (i.e. they have kept it the same or increased it) should not be rewarded. The issues surrounding this problem are complex and require further research. We have, however, developed a heuristic for rewarding agents which functions very well within the optimization framework. This method first finds the current probabilities of selection of the agents, and if the agent team has reduced the objective function value then they are rewarded in proportion of their previous probabilities. This is based on the assumption that an agent whose probability of selection is higher will improve the objective function value more than an agent whose probability of selection is lower.

Table 2 Comparison of all algorithms

	Seq.	Eval.	Min.	Max.	Std. Dev.	Avg.
GHC Alg.	1	10000	1919.3	1927.0	2.2	1921.7
	2	10000	2238.4	2244.2	1.1	2238.6
	3	10000	2245.5	2282.7	19.1	2264.3
Base Alg.	1	1500	1919.3	1935.6	3.5	1920.6
	2	1500	2238.4	2244.2	1.7	2238.9
	3	2500	2245.5	2282.0	10.3	2251.2
Mod. 1	1	1500	1919.3	1928.6	2.2	1920.0
	2	1500	2238.4	2244.2	1.0	2238.5
	3	2500	2245.5	2266.4	5.8	2247.3
Mod. 2	1	1500	1919.3	1925.0	1.5	1920.1
	2	1500	2238.4	2244.2	1.0	2238.5
	3	2500	2245.5	2266.4	8.4	2250.4
Mod. 3	1	1500	1919.3	1923.1	1.2	1919.9
	2	1500	2238.4	2244.2	1.0	2238.5
	3	2500	2245.5	2270.0	8.4	2250.0
Mod. 4	1	1500	1919.3	1925.0	2.1	1920.5
	2	1500	2238.4	2244.2	1.4	2238.7
	3	2500	2245.5	2263.4	7.7	2249.6
Mod. 5	1	1500	1919.3	1927.0	2.1	1920.2
	2	1500	2238.4	2244.2	1.0	2238.5
	3	2500	2245.5	2266.4	4.9	2246.7
Agent Based parameter Alg. with all mods.	1	1500	1919.3	1919.3	0.0	1919.3
	2	1500	2238.4	2238.4	0.0	2238.4
	3	2500	2245.5	2263.4	3.2	2246.0

6.2 Complete Agent-Based Parameter Optimization Algorithm. In this section, the complete agent based parameter optimization algorithm is applied with all the modifications including the modified Hustin move set. The algorithm is shown in Fig. 5.

Once the population is generated randomly and sorted then the number of agents to be called is decided by Modifications 2 and 3, which call the agents depending upon the location of the popula-

1. Enter the population size and the no. of generations (gen).
2. A priori select the appropriate instantiation agents associated with the process sequence.
3. Decide the probability of selection of the instantiation agents (prorata basis).
4. Generate initial population.
5. I= 1
6. While (I <= gen)
   - Sort the population in increasing order of cost;
   - Categorize the population into Good (top 50%) and Poor (remaining 50%);
   - Reject Poor;
   - **Select the no. of agents to be applied based on Modification 2 and 3;**
   - **Select the agents based on the current probabilities;**
   - **Apply the agents on the population member;**
   - **Update the probability of selection of agents through modified Hustin move set (Modification 1 with multi agent heuristics), positive feedback history (Modification4) and negative feedback (Modification 5);**
   - I=I+1.

Fig. 5 Agent-based parameter optimization algorithm for a given sequence of processes

tion member relative to the others; more agents are selected for poor population members. Then, the agents are called based upon their current probabilities of selection. Once the agents are called on a population member to generate a new member then the objective function value of the new member is compared with the parent member to see whether there is success (the objective function value is reduced) or a failure (the objective function value is increased or remains the same). Depending upon the success or failure of the agents, the agent probabilities are updated by the modified Hustin move set (Modification 1 with multi agent heuristics), positive feedback history (Modification 4) and negative feedback (Modification 5).

The updating of probabilities of the agents is done by considering the contribution of Modifications 1, 4 and 5, which translates into the following formula:

Let P_i = Probability of selection of i^{th} agent.

$$P_i = P_u * P_{\text{initial proportion},i} + P_{\text{quality factor}} * \frac{Q_i}{Q_{\text{total}}} + P_{\text{positive feedback}} * \frac{P_{s,i}}{P_{\text{total}}} - P_{\text{negative feedback}} * \frac{P_{f,i}}{P_{f,\text{total}}},$$

with the constraint:

$$P_u + P_{\text{quality factor}} + P_{\text{positive feedback}} - P_{\text{negative feedback}} = 1.$$

Here, $P_{\text{initial proportion},i}$ is the initial prorata probability of selection of agent and P_u is the weight given to it, $P_{\text{quality factor}}$ is the weight given to quality factors calculated using the modified Hustin move set (Sec. 5.1, Modification 1), $P_{\text{positive feedback}}$ is the weight given to positive feedback history calculated using the method presented in Sec. 5.4 (Modification 4) and $P_{\text{negative feedback}}$ is the weight given to negative feedback using the method presented in Sec. 5.5 (Modification 5).

The results from the base case, base case along with individual modifications and the agent based parameter optimization algorithm with all modifications are presented in Table 2.

For sequence 1 the results at 1500 evaluations (30 generations) are presented as the algorithm with individual modifications converged fully at 2500 evaluations (50 generations). Observe that for sequence 1 at 1500 evaluations the complete algorithm with all modifications converged fully. Also shown are the results from the algorithms in which individual modifications were done, none of which converged completely. For sequence 2, all of the algorithms converged fully at 2500 evaluations (50 generations) and so comparison was done at 1500 evaluations (30 generations) at which the complete algorithm with all modifications completely converged. We also show the results from the previous individual modifications, none of which reach full convergence. For sequence 3, at 2500 evaluations the complete algorithm with all modifications had the lowest standard deviation and average among all previous algorithms.

Additionally, the first row in Table 2 presents the results from GHC Algorithm for the three process sequences after 10000 evaluations. Observe that the complete agent based optimization algorithm has lower maximum, standard deviation and average compared to the GHC algorithm at 1500 evaluations for process sequences 1 and 2 and 2500 evaluations for process sequence 3. Thus, the complete agent based algorithm gives better results in less than one-fourth of the evaluations of the GHC algorithms.

The algorithm was run for 30 different random number seeds for all the three process sequences. For some seeds, the algorithm is able to find the minimum quite quickly. For process sequence 1, the optimum value was obtained from one random number seed after only 5 generations (350 iterations); for process sequences 2 & 3, the optimum value was obtained after only 2 generations (200 iterations). This shows that the algorithm is able to find the optimum during the initial stages of the iterative process itself.

Table 3 Agent based parameter algorithm with all modifications applied to continuous problem

	Min.	Max.	Std. Dev.	Avg.
Sequence 1	1761.2	1933.8	57.0	1837.8
Sequence 2	2021.9	2054.1	8.7	2033.2
Sequence 3	2081.0	2359.6	49.2	2132.2

However, for the algorithm to consistently reach, or get close to, the optimum from all random number seeds a greater number of iterations are required.

7 Extension to Continuous Parameter Problem

To this point in the paper we have assumed a problem representation in which the various parameters can only take on discrete values. This assumption, used in the literature to enable formal analysis of optimality, overestimates the optimum. To remove this deficiency we changed the agent definition such that agents can select any real number within a given range. Then we applied the complete agent-based optimization algorithm (with all modification as shown in Sec. 6) to the process sequences and ran it for 200 generations (50 designs×200 generations ≅10000 evaluations). Because the agents now have an infinite number of choices for parameter values, as opposed to the discrete values used above, the algorithm requires a longer run time. We chose to run it as long as the original GHC algorithm ran (10,000 evaluation). As observed in the results (Table 3), the algorithm reduces the minimum in all three sequences substantially:

Sequence 1: 1919.3 to 1761.2,
Sequence 2: 2238.4 to 2021.9,
Sequence 3: 2245.5 to 2081.0.

The byproduct of this approach however, is that the search space become considerably larger. As a result, the standard deviation is higher compared to the discrete optimization problem even after 10000 evaluations.

8 Integrated Agent Based Configuration Optimization Algorithm

The Agent based parameter method has been shown to be an effective technique for process parameter optimization for a given process sequence. In this section the Agent based parameter method is extended and modified for process sequence optimization. Since process sequence optimization includes process parameter optimization as well, the same instantiation agents developed for parameter optimization are used. Similarly, the concept of having population of designs is also used, however the population is different in not only process parameters but also process sequences. Additionally, we use the same selection criteria by keeping the Good population members and rejecting the Poor members and by using the Good members to replace the Poor members by application of agents. Also, the techniques of modified Hustin move set, positive feedback and negative feedback are also used. The integrated Agent based configuration algorithm presented here is different in that it has a set of new sequence agents for modification of sequences.

The integrated Agent based configuration algorithm is presented in Fig. 6 with extensions for process sequence optimization highlighted. The algorithm uses two types of agents: instantiation agents for process parameter modification and sequence agents for modifying a sequence. The algorithm begins by forming an initial population of candidate designs. Each population member can be different not only in terms of process parameters but also in terms of process sequences. The algorithm proceeds as above. However, if the objective function value is not reduced then the sequence of the design is modified by the sequence agents. There are three types of sequence agents used (explained in more detail in the

1. Enter the population size and the number of generations (gen) and the number of iterations
2. Generate initial population
   a. **Generate the initial process sequences**
   b. Generate the associated process parameters
3. I=1;
4. While (I<= Gen)
   a. Sort the population in increasing order of cost
   b. Categorize the population into Good (top 50%) and poor (remaining 50%).
   c. Reject poor.
      Select the no. of instantiation agents to be applied;
   d. Select the instantiation agents based on current probabilities;
   e. Generate new members by modifying the parameters by applying instantiation agents. **If no improvement results then the sequence is modified by applying the sequence agents, thus changing the sequence. During the initial generations sequence agents are called more often compared to later generations.**
   f. Update the probability of selection of instantiation agents through modified Hustin move set, positive feedback and negative feedback;
   g. Update the probability of sequence agents
   h. I=I+1;

Fig. 6 Integrated Agent based Configuration Algorithm for process sequence and process parameter optimization

following section): process addition sequence agent, process removal sequence agent and process restarting sequence agent. Depending upon how each sequence agent performs, whether it reduces or increases the objective function value, its probability of selection is correspondingly increased or decreased after each generation. The number of sequence agents called is based on a linear curve with more number of sequence agents called during the initial stages of the iterative process than later stages to ensure exploration of large number of sequences initially and then shift focus to optimizing the parameters of those sequences during later stages. The process continues until it reaches the specified limit on the number of generations or the number of iterations.

9 Sequence Modification Using Sequence Modification Agents

The sequence modification agents ensure that the algorithm starts with and maintains a set of feasible process sequences. The feasible sequences are formed by starting from a sequence of casting and finish machining processes and then adding other processes in between using a simple pair-wise constraint satisfaction. The procedure uses constraints from the limitations of the manufacturing processes by comparing the geometry of the output of a process to the input geometry of the next process. There are two types of geometries possible: cylindrical (c) and general (g) (which includes cylindrical). The constraints are listed in Fig. 7 that shows the valid and invalid ways in which the output geometry of one process can be followed by the input geometry of the

Output shape of process	Input shape to next process	Result
Cylindrical(c)	Cylindrical(c)	Valid
Cylindrical(c)	General(g)	Valid
General(g)	Cylindrical(c)	Invalid
General (g)	General(g)	valid

Fig. 7 Constraints on geometries of consecutive processes

No.	Process	Input Shape of process	Output Shape of process
1	Casting	No shape: molten metal(x)	Cylindrical(c)
2	Upset Forging	Cylindrical(c)	Cylindrical(c)
3	Blocker Forging	Cylindrical(c)	General(g)
4	Extrusion	Cylindrical(c)	Cylindrical(c)
5	Preform Machining	Cylindrical(c)	Cylindrical(c)
6	Rough Machining	General(g)	General(g)
7	Finish Machining	General(g)	General(g)

Fig. 8 Restrictions on input and output geometries of process

next process. Only a general shape followed by a cylindrical shape is not feasible since only in very special circumstances would a general shape be a cylindrical shape. Figure 8 lists the restrictions on input and output geometries of various processes. Consider the process casting(x,c) (the symbol "x" is used to denote a molten metal) followed by upset forging(c,c) following an (input shape, output shape) nomenclature. The output of casting is a cylindrical workpiece and the input of upset forging should be a cylindrical workpiece, hence this pair casting $\rightarrow$upset forging is feasible. However, blocker forging(c,g) followed by blocker forging(c,g) would be infeasible based on our current model because the output from blocker forging is a general shape whereas the input to blocker forging process should be cylindrical.

Based on this pair-wise comparison only feasible sequences are formed. Furthermore, a sequence can be modified in three different ways: addition of a process, removal of a process or restarting the generation of a sequence. For each of these sequence modifications there is a corresponding sequence agent which does the modification: Process addition sequence agent (for adding a process), Process removal sequence agent (for removing a sequence), Process restarting sequence agent (for restarting a sequence). Each of these agents is explained in more detail below:

Process Addition Sequence Agent. Before a process is added the position in the process sequence at which it is to be added is determined by generating a random number between zero and the sequence length. Then during addition of the process it is ensured that the input of the process matches the output of the process (which is guaranteed) after which it is added. However, it is not necessary that the output of the process to be added matches the input of the next process. As a result more processes might need to be added to make a sequence feasible. This process continues until the sequence size reaches a limit, here 10, which is the maximum sequence size. If the sequence size reaches the maximum, it is removed and a new sequence is generated with casting and finish machining as the initial and final process respectively in the sequence.

Process Removal Sequence Agent. For removal of a process a position is selected for the process to be removed from a sequence. The position is determined by generating a random number between zero and the sequence length. Once a process is removed it might leave the remaining sequence infeasible as the constraints listed in Fig. 7 might be violated. To make the sequence again feasible processes are added at the position that are making it infeasible. This process continues until a feasible sequence is obtained or the maximum size limit is reached. If the maximum number is reached the sequence is removed and a new one built.

Process Restarting Sequence Agent. The sequence generation process is restarted when the sequence size becomes more than a limit, in this work 10. The reason is that long sequences involve additional set up costs for each process thus making them costly compared to shorter sequences. Hence, a limit of size 10 is kept.

Table 4 Optimization result after 20000 iterations for 30 random number seeds

Minimum	Maximum	Std. Deviation	Average
1919.3	2250.1	154.5	2069.8

9.1 Selection of Sequence Agent. Each of these sequence agents is given an initial uniform probability of selection. Then depending upon how an agent has performed (whether it has increased or reduced the objective function value) its probability of selection is increased or decreased. Formally,

Let S_i=Probability of selection of i^{th} sequence agent and $S_{\text{initial proportion},i}$=initial uniform probability of selection of i^{th} sequence agent.

Given:

$$S_{s,i} = \frac{\text{No. of successes of } i^{th} \text{ sequence agent throughout the generations}}{\text{Total number of times the sequence agent has been applied}},$$

$S_{s,\text{total}}=\Sigma_{i=1}^{n} S_{s,i}$, where n=total number of agents,

S_u=Proportion of uniform probability (determines the trade-off of uniform probability with positive feedback),

then

$$S_i = S_{\text{initial proportion},i} * S_u + (1-S_u) * \frac{S_{s,i}}{S_{s,\text{total}}}$$

As shown above S_i is formed from two terms: $S_{\text{initial proportion},i}$ and $S_{s,i}/S_{s,\text{total}}$ which comes from the proportion of the number of times the agent has been successful in the previous generations, with S_u $(0<S_u<1)$ deciding the relative weight of each.

10 Results for Configuration Optimization

The input to the integrated configuration algorithm was the geometry and material data of the turbine disk. The algorithm then uses the method outlined in Fig. 6 for finding the optimal sequence with the associated optimal process parameters.

The algorithm was run for 30 different random number seeds. For each random number seed the algorithm was run for 20000 iterations. 20000 iterations were chosen because the search space for process sequence optimization is very large compared to the process parameter optimization problem, for which the algorithm was run for 10000 iterations. The result in Table 4 shows the minimum, maximum, standard deviation and average of the solution after 20000 iterations for 30 random number seeds.

The minimum was obtained for the following sequence:
Casting→Upset Forging→Rough Machining
→Finish Machining.
This sequence and minimum objective function value is one of the three sequences explored in Jacobson et al. [5] and above for the parameter optimization problem. Hence, this approach shows that it is the optimum/near optimum sequence out of the space of all feasible sequences.

10.1 Result with Modified Cost function. Typical bulk manufacturing process sequences involve the use of multiple processes of the same type. For example, closed die forging would be used more than once. One reason could be limitations on the available input workpiece from casting; for example, if the initial aspect ratio of the billet from casting is high then it needs to be compressed multiple times using upset forging before any other processes can be used. Another reason is that it may not be possible or at least optimal to get the required material properties by using a process just once.

Although the algorithm presented in this work can adequately handle multiple processes of the same type, the calculation of process cost in the present model cannot handle such cases. Thus we now modify the cost function used in forging by adding an additional term to represent the fact that the cost of forging increases as the amount of compression performed increases. This is represented by the following term:

$$\text{Cost\_compression} = K\left[\frac{\text{Final Height}}{\text{Initial Height}}\right]^2, \text{ where } K=\text{constant}.$$

This additional term is introduced in upset and blocker forging.

Table 5 Optimization result after 20000 iterations for 30 random number seeds with modified cost function and constant casting geometry.

Minimum	Maximum	Std. Deviation	Average
7189.1	7311.3	35.6	7223.2

The algorithm with this modified cost function and a constant casting geometry (radius and height were constant, with radius = 3.0 inches and height = 8.0 inches) was run for 20000 iterations and 30 different random number seeds. Table 5 shows the result of applying the algorithm.

The minimum was obtained for the following sequence:
Casting→Upset Forging→Upset Forging→Rough Machining
→Finish Machining.
Note that upset forging was used two times in the same sequence implying that it is more cost effective to do the upset forging operations in two steps rather than a single step. This validates our approach to explore sequences with multiple processes of the same type.

11 Conclusions

This paper introduces an agent-based optimization algorithm that combines deterministic and stochastic optimization search strategies. A specific application to the bulk manufacturing process planning problem is presented and proven to be quite effective. In this algorithm the parameter agents are modeled on manufacturing processes and contain domain knowledge specific to bulk manufacturing. The agents then develop a population of designs and modify it in an interactive framework. Configuration agents build up and introduce changes to the configuration sequence, there enabling sequence optimization.

The agent based optimization is based on domain knowledge and a stochastic search process. Incorporating domain knowledge helps to prune down the search space and drives the search toward optima, whereas the stochastic search process working in a complementary fashion ensures that a variety of designs are generated with higher likelihood of success. Although the parameter optimization agents are somewhat simplistic in this application, they both show the power of this approach and allow for more complex agents in other applications. The configuration agents, however, are both more complex here and function within a much more complex topology optimization process.

The base parameter optimization algorithm has been shown to be more effective than the results found in the literature from an efficiency viewpoint. Modifications made to the base algorithm were tested on the process sequences. Each modification improved the performance of the algorithm. Following this, all the modifications were combined into one complete parameter optimization algorithm and applied to individual process sequences, with even superior performance.

Finally this algorithm was extended to develop an integrated approach for performing manufacturing process optimization by simultaneously performing process sequence and process parameter optimization. By using this method a large number of sequences are explored leading to a potentially better solution. First the technique is implemented using a previously presented cost function. The result converges to one of the sequences found in the parameter optimization work. However, when the cost func-

tion is modified to better represent forging operations, then a new (previously unknown) sequence with multiple processes is found to be the optimum. This validates our current approach of performing combined process sequence and process parameter optimization.

Acknowledgments

The authors would like to thank Garth Frazier of WPAFB and Enrique Medina of Austral Engineering Services for helpful discussions and Sheldon Jacobson of University of Illinois, Urbana Champaign for providing the GHC code. This research effort was sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant numbers F49620-98-1-0172 and F49620-01-1-0050. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or the U.S. Government.

References

[1] Yin, S., 2000, "*A Computational Framework for Automated Product Layout Synthesis Based on an Extended Pattern Search Algorithm*," Ph.D, Dissertation, Carnegie Mellon University, Pittsburgh, PA.

[2] Fischer, C. E., Gunasekera, J., and Malas, J. C., 1997, "Process Model Development for Optimization of Forged Disk Manufacturing Processes," Steel Forging: Second Volume, ASTM STP 1259 (E. G. Nisbett and A. S. Melilli, eds.), American Society for Testing and Materials.

[3] Gunasekera, J. S., Fischer, C. E., Malas, J. C., Mullins, W. M., and Yang, M. S., 1996, "Development of Process Models for Use With Global Optimization of a Manufacturing System," *Proceedings of the ASME Symposium on Modeling, Simulation and Control of Metal Processing*, ASME International Mechanical Engineering Congress, Atlanta, GA, November.

[4] Chang, T. C., and Wysk, R. A., 1985, *An Introduction to Automated Process Planning*, Prentice-Hall, Englewood Cliffs, New Jersey.

[5] Jacobson, S. H., Sullivan, K. A., and Johnson, A. W., 1998, "Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms," Eng. Optimiz., **31**, pp. 247–260.

[6] Vaughan, D. E., Jacobson, S. H., and Armstrong, D. E., 2000, "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms," Trans. ASME, **122**, pp. 164–171.

[7] Storr, A., Li, R., and Stroehle, H., 2000, "Agent-based NC Process Planning for Complete Machining on Turning Centers," *Proceedings of the 2000 ASME DETC and CIE Conference*, DETC2000/CIE-14635, Baltimore, Maryland, September 10–13.

[8] Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan.

[9] Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," Science, **220**(4598), pp. 671–679.

[10] Hustin, S., and Sangiovanni-Vincentelli, A., 1987, "TIM, a New Standard Cell Placement Program Based on the Simulated Annealing Algorithm," *IEEE Physical Design Workshop on Placement and Floorplanning*, Hilton Head, SC, April.

[11] Campbell, M., 2000, "*The A-Design Invention Machine: A Means of Automating and Investigating Conceptual Design*," Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

[12] Campbell, M., Cagan, J., and Kotovsky, K., 1999, "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment," Res. Eng. Des., **11**, pp. 172–192.

[13] Langton, C. G., ed., 1988, *Artificial Life*, Addison Wesley, Reading, MA.

[14] Russell, S., and Norvig, P., 1995, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

[15] Mori, T., and Cutkosky, M. R., 1998, "Agent-based Collaborative Design of Parts in Assembly," DETC1998/CIE-5697, *Proceedings of the 1998 ASME DETC*, Atlanta, Georgia, September 10–13.