Search Strategies in Evolutionary Multi-Agent Systems: The Effect of Cooperation and Reward on Solution Quality

Lindsay Hanna Landry
e-mail: lindsay.h.landry@gmail.com

Jonathan Cagan[1]
e-mail: jcag@andrew.cmu.edu

Department of Mechanical Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213

Cooperation and reward of strategic agents in an evolutionary optimization framework is explored in order to better solve engineering design problems. Agents in this Evolutionary Multi-Agent Systems (EMAS) framework rely on one another to better their performance, but also vie for the opportunity to reproduce. The level of cooperation and reward is varied by altering the amount of interaction between agents and the fitness function describing their evolution. The effect of each variable is measured using the problem objective function as a metric. Increasing the amount of cooperation in the evolving team is shown to lead to improved performance for several multimodal and complex numerical optimization and three-dimensional layout problems. However, fitness functions that utilize team-based rewards are found to be inferior to those that reward on an individual basis. The performance trends for different fitness functions and levels of cooperation remain when EMAS is applied to the more complex problem of three-dimensional packing as well. [DOI: 10.1115/1.4004192]

1 Introduction

The performance of algorithms for solving complex design problems is often closely tied to parameter settings, initial conditions, and problem instance [1–4]. It can therefore be very difficult to decide, given any particular problem, what algorithm and parameters are best suited to solve it. One option, the basis of algorithm portfolios [5], is to have multiple algorithms available and learn on a test set of solutions how to optimize the application of each. Another option, and the basis of this work, is to embody strategies (algorithms and parameters) for solving the problem in autonomous agents and allow those agents to cooperate by sharing solutions [6,7]. An agent in this case may be defined as an entity capable of autonomously sensing, processing, and effecting changes to its environment (either physical or software) [8].

The cooperation of agents representing strategies and capabilities grouped together in multi-agent systems (MAS) has been shown to be very successful in solving engineering design problems in systems such as A-Teams [9], A-Design [10], and blackboard systems [11]. Synergy (i.e., multiple strategies working cooperatively to provide better results than any single strategy) is an emergent property of these systems [7]. Even with imperfect algorithm and parameter settings, superior solutions can thus be achieved by introducing cooperation. However, the algorithms and parameters employed by agents in these approaches typically remain static over the course of the optimization. For example, in A-Design [10], the strategies used by creator and modifier agents do not change over the course of the entire algorithm. As a result, the full potential of each strategy is not explored or adapted to the problem at hand.

A synergistic MAS approach entitled Evolutionary Multi-Agent Systems (EMAS) was developed to combat this issue [12,13]. The EMAS framework is defined by a group of agents, each with a unique chromosome describing a solution creation or modification strategy, sharing and refining solutions while the group is subjected to genetic operators for mutation, recombination, and selection. In this way, the agent space (i.e., the strategy space) is explored in tandem with the solution space. EMAS is similar to MAS approaches like A-Teams and A-Design [9,10] in that agents cooperate by sharing solutions in a flat (nonhierarchical) organization. Unlike other MAS approaches; however, each agent's skills (strategies) are not fully defined at the start of the optimization. Instead, the agents' strategies change over time as the group of agents is subjected to evolution. Evolutionary Multi-Agent Systems represents a unique extension to the area of evolutionary computing in that this evolution occurs in tandem with the cooperation between the agents (the individuals in the evolving population) and the search for a solution. Frameworks for cooperative co-evolution utilize cooperation between related *subpopulations* [14], rather than between individuals in the *same* population. Perhaps the closest analogy is that of the island model genetic algorithm in which populations are subjected to different crossover and mutation operators [3]. In this case, the transferring of solutions between subpopulations could be considered cooperation, though unlike EMAS no two strategies may be applied to the same population of solutions.

In the past, EMAS has been applied to the traveling salesman problem (TSP), in which the goal is to find the lowest cost round-trip path that visits a set of cities. It was demonstrated that the evolving cooperative agent team was more successful at producing high quality solutions than either individual agents acting independently or randomly designed teams working cooperatively [12,13]. Evolution of the agents was shown to be critical to the success of the team. In addition, it was found that the pattern of agent evolution (the characteristics of the agent population over time) deduced from several EMAS runs on a 48-city TSP problem could be directly applied to obtain good results for a 532-city problem without the burden of evolutionary learning. These findings suggest that EMAS is capable of cooperatively and

[1]Corresponding author.

Contributed by the Design Theory and Methodology Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received October 19, 2009; final manuscript received April 20, 2011; published online June 15, 2011. Assoc. Editor: Yan Jin.

adaptively searching the strategy space to apply the most effective set of algorithms at the point in the search when they will be most helpful.

Because EMAS is evolutionary, a key ingredient to its definition is the agent fitness function, or how the agents are evaluated when it comes time to reproduce. The fact that individuals in the evolving population (agents) cooperate in EMAS makes credit assignment for those individuals more challenging than in traditional genetic algorithms. For instance, consider that some agents in the team may create solutions that do not have good objective function values but are still good starter solutions for other agents. If these helpful agents are not recognized in the fitness function, they may not survive reproduction and the team as a whole could suffer from their loss. Studies of human organizations have shown that the level of cooperation and interaction between individuals in an organization is central to determining the most appropriate way to compensate them [15,16].

While the fitness function of the agents and the amount of cooperation between them are linked in EMAS, the relationship between the two was not a focus of the initial study. Furthermore, while this previous work established that the team outperformed individuals, the effectiveness of cooperative versus noncooperative groups was not explicitly studied. *The goal of this work is to better understand the fitness function/cooperation relationship by varying the level of cooperation between the agents as well as the fitness function for their adaptation.* The level of cooperation is varied by modifying the agents' ability to share solutions. In addition, several fitness functions are explored which vary in terms of the reward given to agents for different contributions to the shared solution pool. Several numerical optimization and three-dimensional (3D) packing problem instances provide the test bed for this examination.

2 Evolutionary Multi-Agent Systems

2.1 Framework. A graphical representation of the EMAS framework is shown in Fig. 1. A single iteration of EMAS consists of each agent in the team running its algorithm on an initial solution. For this study, there are 8 agents in the team which run for 25 iterations. Agents run their algorithms in parallel to maximize the efficiency of the framework. The initial solution used by an agent is determined by the cooperation protocol and is either taken from the shared memory[2] (as indicated by dashed lines in Fig. 1) or is randomly generated from the solution space (dotted lines) (see Sec. 2). The output of each agent's algorithm is then automatically sent to the shared memory (solid lines). All solutions in the memory are thus the by-product of agents running their algorithms. To keep the size of the memory in check, some solutions are removed from it every time the agents are subjected to reproduction. Solutions are removed based on (1) their objective function value and (2) whether or not agents have used them as input in the past. A multi-objective analysis is conducted using these two criteria and solutions that are dominated (not on the Pareto front) are removed from the memory first. If only Pareto solutions exist in the memory but the number of solutions is still too great, solutions are removed randomly. This destruction protocol differs from that used in the original EMAS study, which only used solution quality as a metric for destroying solutions. It is believed that also using the number of times a solution has been used successfully will help maintain diversity in the shared memory.

2.2 Cooperation. Cooperation between the agents is varied by changing how the initial solution used by an agent is determined. This is in contrast to collaboration, in which agents' decisions are coupled [17]. When an agent uses a solution from the memory as the input for its algorithm, that agent is considered to be cooperating with the rest of the team because it is building off of the work of other agents. If an agent instead uses a random initial solution as its input, it is considered to be not cooperating with its teammates. In the original framework [12], agents always used solutions from a shared memory, so the agents cooperated fully. In this study, a fully cooperative scenario, like the one used in the original framework, as well as partially cooperative and noncooperative scenarios are examined. The three levels of cooperation examined in this study are defined as follows (refer to Fig. 1).

(1) Full cooperation: Each agent will always use an initial solution from the shared memory as the input to its algorithm (dashed lines only). Only this level of cooperation was considered in the original study.
(2) Partial cooperation: Each agent has a 50% chance of using a randomly generated initial solution or an initial solution from the shared memory as the input to its algorithm (dashed or dotted lines).
(3) No cooperation: Each agent will always use a randomly generated initial solution as the input to its algorithm (dotted lines only).

Note that in the partial and full cooperation scenarios, it is possible that an agent chooses a solution from the memory that it created in a previous iteration. In this case, it is building off of its own work, rather than building off of the work of others. While this is not in the spirit of cooperation as is defined here, it is likely that both the method of destroying solutions in the memory (see Sec. 1) and the randomness with which solutions are chosen from the memory prevent the frequent occurrence of this phenomenon.

The no cooperation scenario can be likened to traditional genetic programming (GP), where algorithms are evaluated based on their ability to generate high quality solutions to a given problem. In this situation, *solution* information (as opposed to the *algorithm* information that is typically sought in GP) is lost after each generation of algorithms, because the next generation attempts to solve the same problem without using the final solutions of the previous generation. In essence, each new generation of algorithms represents a "random restart" in the overall search for the optimum solution. In the partially and fully cooperative

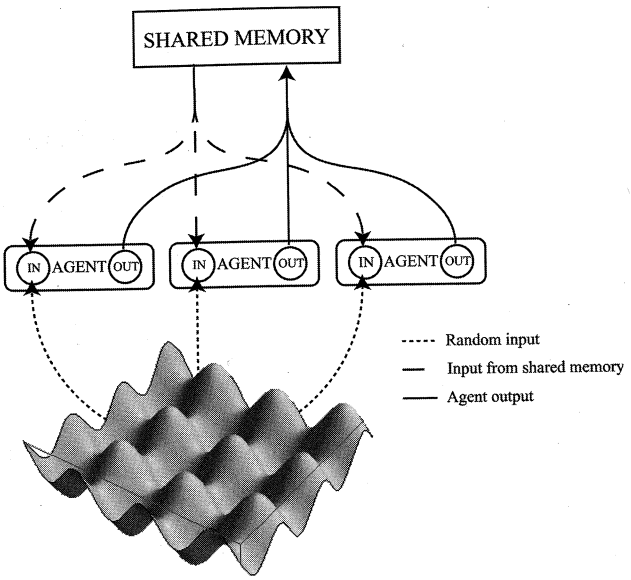


Fig. 1 Graphical representation of EMAS implementation for a single iteration. Lines represent the flow of solutions from the shared memory (top) to the agents (middle), from the solution space (bottom) to the agents, and from the agents to the shared memory.

[2]In the original version of EMAS [12], there were multiple shared memories and an agent's chromosome identified which memory it would use. With multiple memories, cooperation is not guaranteed between all agents, so to isolate and examine the effect of cooperation a single memory is used here.

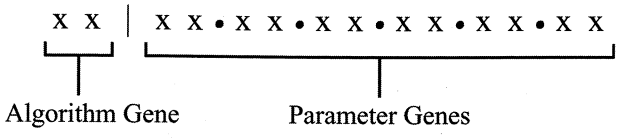


Fig. 2 EMAS agent chromosome representation

scenarios, solution information is not lost from one generation of algorithms to the next. However, this does not necessarily guarantee that final solution quality will be enhanced, especially in highly multimodal problem spaces. For such spaces, it might be argued that keeping track of the best solution and performing random restarts of ever-evolving algorithms would be at least as effective as relying solely on the solution output of previous algorithms (as in the fully cooperative scenario). In the noncooperative scenario, agents' input solutions are not restricted to any area of the solution space, so agents may potentially explore more of that solution space. In cooperative scenarios; however, input solutions are limited to those uncovered by other agents, which may point to local minima.

2.3 Agent Representation. The strategy (algorithm and parameters) employed by each agent in EMAS is described in its own unique chromosome. Recall that evolution in EMAS takes place at the *agent* level, meaning that agents themselves evolve over time as they apply their algorithms to solutions. Agent chromosomes are binary with a single gene describing the algorithm type and additional genes for parameter definitions. Figure 2 is a template for the agents' chromosomes. Each gene in an agent's chromosome consists of two binary bits, which means that there are $2^2 = 4$ different values associated with it.

The algorithm gene, separated in Fig. 2 from the parameter genes by a vertical bar, determines whether the agent will run pattern search, a genetic algorithm, simulated annealing, or Powell's method. These algorithms are defined as follows.

(1) Pattern search [18–20]: Steps are taken from an initial solution along pattern directions corresponding to the positive and negative univariate search directions. If the step results in a better solution, that solution is accepted as the starting point for the next step; otherwise it is rejected and the algorithm reverts to the previous solution. If no steps are accepted along any of the pattern directions, the step size is decreased by a scale factor and the patterns are explored again with the new step size. This process is repeated until the step size falls below a tolerance value.
(2) Genetic algorithm [21–23]: Solutions are encoded as binary chromosomes with a single gene of a specified length representing each coordinate of each component. A population of these solutions is subjected to single-point crossover, random mutation, and roulette-wheel selection to create a new population. This process is repeated until the number of generations (population reproductions) has reached the specified amount.
(3) Simulated annealing [24–28]: Starting with an initial solution, a new solution is generated that corresponds to the end point of a vector originating from the initial solution with a magnitude equal to one of several prespecified step sizes and a randomly generated direction. The probability of any particular step size is adjusted as the algorithm progresses according to the procedure given in Hustin and Sangiovanni-Vincentelli [24]. If this new solution is better than the previous solution, it is accepted as the new starting point. Otherwise, the solution may still be accepted with some probability. The probability of accepting a worse solution is dependent on a temperature parameter, which decreases over the course of the optimization according to the procedure outlined in Huang et al. [25].
(4) Powell's method [29]: The bounded one-dimensional Golden Section method is used to search along univariate directions. In subsequent iterations, Golden Section is used to search along conjugate directions created by connecting previously found optimal points. Powell's method is a downhill search method that is typically applied to unimodal numerical optimization problems.

Though there are many other successful algorithms and strategies developed for numerical optimization as well as 3D layout (e.g., Fadel et al. [30] and Tiwari et al. [31]), the algorithms chosen are representative of the different categories of search: direct, evolutionary, stochastic, and downhill, respectively.

Associated with each algorithm are parameters that describe how the search will be conducted. These parameters are also encoded in the agent's chromosome as illustrated in Fig. 2 (each parameter gene is separated by a center dot). Each gene is made up of two bits, meaning that there are four possible values for each parameter. The parameters and values associated with each algorithm are given in Table 1. Note that some algorithms have more parameters than others, though all agents have six parameter

Table 1 Algorithms and parameters used for EMAS implementation on numerical optimization and 3D packing

		Values			
Algorithm	Parameter	00	01	10	11
Pattern search	Initial step size (fraction of search space)	0.1	0.3667	0.6333	0.9
	Tolerance (fraction of search space)	0.0001	0.01	0.3	0.9
	Scale factor	0.1	0.3667	0.6333	0.9
Genetic algorithm	Crossover probability	0.75	0.85	0.95	1
	Mutation probability	0.001	0.01	0.1	0.5
	Bits to encode each variable	3	4	5	6
	Number of generations	10	50	100	250
	Population size	10	50	100	250
	Selection pressure	1	1.5	2	2.5
Simulated annealing	Maximum neighborhood size (fraction of search space)	0.1	0.3667	0.6333	0.9
	Minimum neighborhood size (fraction of search space)	0.001	0.01	0.1	0.5
	Number of available step sizes	2	3	5	7
	Maximum number of moves at any temperature	10	100	200	500
	Minimum number of steps before testing for equilibrium	0.1	0.2	0.3	0.4
	Minimum ratio of old to new temperature (to prevent early convergence)	0.5	0.75	0.8	0.85
Powell's method	Initial guess multiplier for Golden Section method	0.01	0.03	0.06	0.1
	Tolerance for Golden Section method	0.00001	0.0001	0.001	0.01
	Number of iterations	1	2	3	4

genes (as shown in Fig. 2). For instance, the genetic algorithm and simulated annealing have six parameters, where pattern search and Powell's method have only three. To address this difference, the last three parameter genes are ignored for pattern search and Powell's method agents. This means that there are $4^6 = 4096$ genetic algorithm and simulated annealing configurations, but only $4^3 = 64$ different pattern search and Powell's method configurations. Thus, while the total number of chromosomes representing simulated annealing is the same as the total number of chromosomes representing genetic algorithms, pattern search, and Powell's method, the number of *different* genetic algorithm and simulated annealing chromosomes is greater than the number of *different* pattern search and Powell's method chromosomes. The parameter values for each gene configuration are designed such that values associated with a less thorough search are coded as 00 and parameter values corresponding to a more thorough search are coded as 11. For instance, tolerances for both pattern search and Powell's method are larger when the associated gene is 00 than when that gene is 11, implying less accurate output in the 00 case. This ordering was done so that even if the algorithm gene is altered through mutation or recombination of the chromosome, the scope of the search (whether narrow or broad) may be maintained. The values given in Table 1 are identical for all problems explored in this study—both numerical optimization and 3D packing. As a result, some of the parameters have values that are given as a fraction of the finite search space. For implementation of pattern search, each of the three parameters describes characteristics of the search along both translational and rotational directions.

2.4 Agent Fitness Function and Reproduction. Reproduction of the agent population occurs every other iteration and consists of choosing pairs of agents to be parents and applying crossover and mutation operators to their binary chromosomes to form children. Crossover is single-point crossover, which means that two children are formed from reciprocal parts of their parents' chromosomes. The probability of performing crossover on any chosen pair of parents is 0.9. The mutation rate for the children resulting from crossover is 0.01 per allele and is applied by changing the allele from 0 to 1 or vice versa. These crossover and mutation probabilities are for the *agent population* and should not be confused with those of the genetic algorithm described in Table 1, which would be applied to *solutions* by an agent with the appropriate chromosome. Once a new generation of agents has been created through reproduction of the previous generation, the new generation replaces the old.

The fitness of an agent determines its likelihood to be chosen as a parent and thus the likelihood that it will pass on its genetic traits. In the original study, an agent's fitness was based solely on its ability to create good solutions. This means that an agent's ability to create helpful solutions was not recognized. For this study, the original fitness function is considered alongside a fitness function that recognizes helpful agents and one which rewards group contributions. The three different fitness functions, labeled individual, assistant, and team, are defined as follows.

(1) Individual: Only those agents that create good solutions (solutions better than the median solution in the memory) will receive a reward. These agents are referred to as *direct contributors*. The individual fitness function is thus given by

$$f_I(a) = \frac{I_a}{T_a} \tag{1}$$

where I_a is the number of solutions created by agent a that are better than the median solution in the memory and T_a is the total number of solutions created by agent a. The individual fitness function was the only function used to evaluate agents in the original framework.

(2) Assistant: Direct contributors receive individual rewards as in the Individual fitness function. However, if a direct contributor improves a solution placed in the memory by another agent, that agent, termed the *assistant*, will receive a reward as well—as long as the assistant is not also a direct contributor. The assistant fitness function is therefore

$$f_A(a) = \frac{I_a + A_a}{T_a} \tag{2}$$

where A_a is the number of solutions created by agent a that were not better than the median solution but were used to create a better solution by another agent.

(3) Team: Direct contributors receive individual rewards as in the Individual fitness function. The rest of the team, termed *noncontributors*, receives a reward equal to the proportion of direct contributors in the team. For example, there are eight agents in the team; if two of them are direct contributors, the six noncontributors would receive a reward of $2/8 = 0.25$. The fitness of an agent under the team fitness function is given by

$$f_T(a) = \frac{I_a}{T_a} + \frac{N_{dc}}{N} \tag{3}$$

where N_{dc} is the number of direct contributors in the population and N is the agent population size.

3 Problem Definition

3.1 Numerical Optimization. Four different two-dimensional numerical optimization (minimization) problems of varying complexity are considered. These and other similar functions also have been used for benchmarking algorithm performance by Campbell et al. [10] and Aladahalli et al. [32]. The first is a Bohachevsky function [see Fig. 3(*a*)] given by

$$g_1(x,y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7 \tag{4}$$

where $-1 \le x \le 1$ and $-1 \le y \le 1$. The second is a sinusoidal function with more significant peaks and valleys but over a larger area than the Bohachevsky function [see Fig. 3(*b*)] and is given by

$$g_2(x,y) = x^2 + y^2 - 100\sin(x) - 100\sin(y) \tag{5}$$

where $-10 \le x \le 10$ and $-10 \le y \le 10$. The third function, shown in Fig. 3(*c*), contains a small concentrated area of large peaks and valleys and is given by

$$\begin{aligned} g_3(x,y) = {} & 300(1-x)^2\exp\left(-x^2\right) - (y+1)^2 - 1000\left(\frac{x}{5} - x^3\right) \\ & - y^5\exp\left(-x^2 - y^2\right) - \frac{100}{3}\exp\left(-(x+1)^2\right) \\ & - y^2 - |x| - |y| \end{aligned} \tag{6}$$

where $-10 \le x \le 10$ and $-10 \le y \le 10$. The fourth function, g_4, is a fractal space generated over the range [0,1] using the diamond-square algorithm, an extension of the random midpoint displacement method described in Fournier et al. [33] and Miller [34] with a depth of recursion of 6, a σ value of 5, and scale factor of 0.1. Fractal spaces are highly multimodal and have been shown to be similar in nature to spaces encountered in other engineering domains like 3D packing [35,19]. When running EMAS on the fractal space, new spaces with the same characteristics (depth of recursion, σ value, etc.) were generated before each run. An example of one such space can be found in Fig. 3(*d*).

3.2 Three-Dimensional Packing. While the numerical optimization problems presented in the last section are useful for benchmarking algorithm performance, higher order optimization problems, specifically those relating to engineering design, should

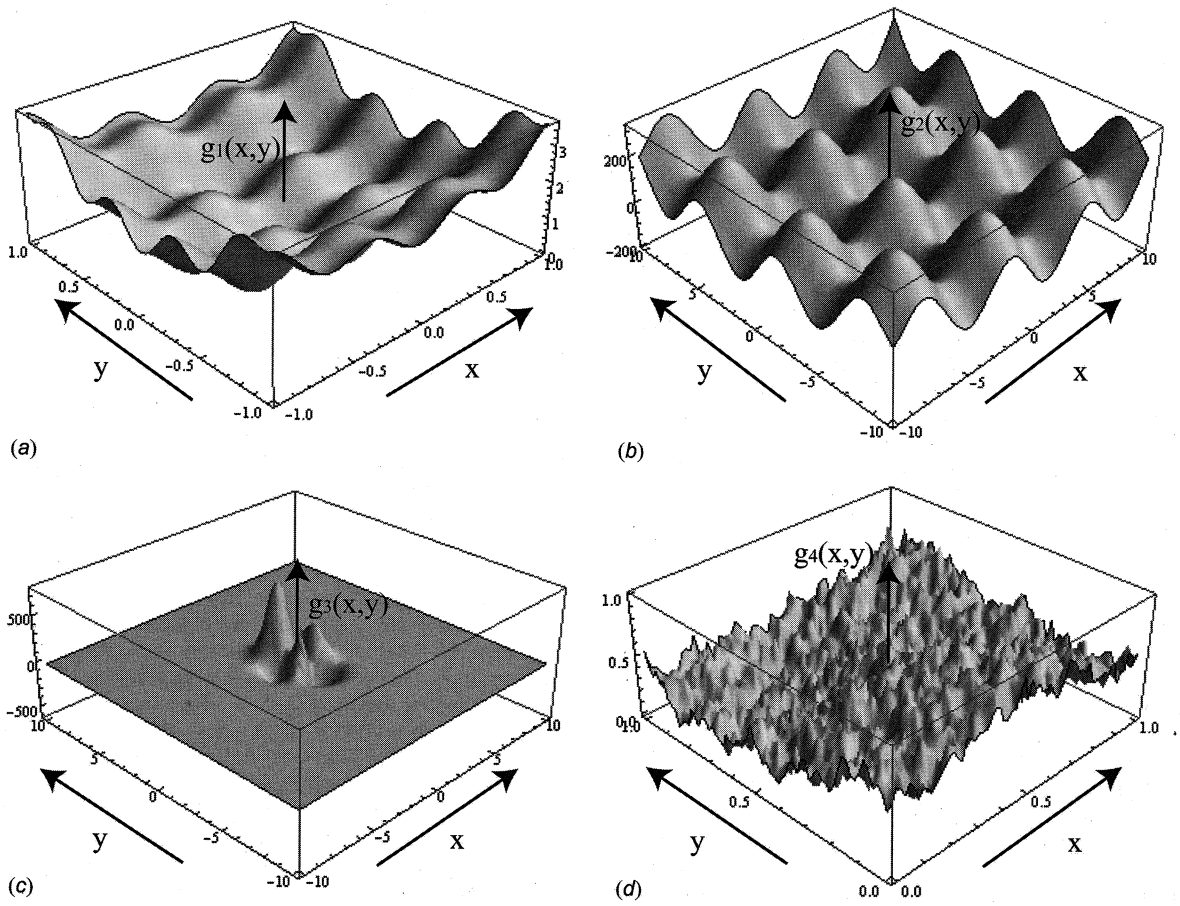


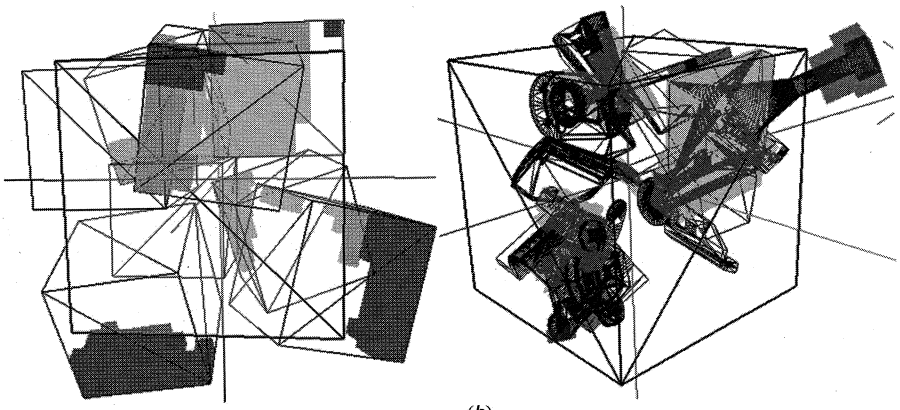
Fig. 3 Function spaces used for studying EMAS for numerical optimization: (*a*) Bohachevsky function, (*b*) sinusoidal function, (*c*) peaks function, and (*d*) fractal function

also be considered when examining potential new algorithms. Thus; in addition to the numerical optimization problems described earlier, the EMAS framework is also applied to solve 3D packing, the problem of placing components in a container without overlap or container protrusion [20]. The 3D packing problem has been extensively studied in the operations research [36,37] and engineering design communities [26,28,30,31] and is highly relevant to design problems such as pallet loading, container and trunk packing, and layout of rapid prototyping wells. It is a very difficult problem and, as stated before, has been shown to have a fractal-like design space, making gradient-based and deterministic methods for solving it impractical. Furthermore, extensions to 3D packing include similarly relevant 3D layout [2,19,20,27,38], which has applications in VLSI layout and routing, facilities layout, and many other areas. Though many problems can be formulated as numerical optimization problems, the ability to solve complex 3D packing illustrates the flexibility of EMAS and its ability to combat relevant design problems.

Two 3D packing problems are considered. The first is a simple eight-cube packing problem, in which eight identical cubes are to be packed inside a cubical container. The side length of each cube is equal to half the side length of the container, so when the cubes are aligned and stacked four on top of four, they exactly fit in the container [see Fig. 4(*a*)]. Though this problem seems trivial for a human problem solver, allowing full rotational and translational freedom makes this a very challenging problem for computational solvers. The second 3D packing problem that is considered consists of 13 components of arbitrary geometry to be packed in a cubical container. These components have holes and protrusions and are the same as those used in Aladahalli et al. [32] [see Fig. 4(*b*)]. The objective function for the 3D packing problem is the intersection and protrusion volume of the components calculated by approximating their geometry with octree data structures [39,28].

4 Results

4.1 Numerical Optimization. The objective function value in the shared memory after 25 iterations of EMAS applied to the four numerical optimization problems discussed in Sec. 1 are provided in Table 2, averaged over 50 independent trials. The number of iterations was set to be 25 based on the fact that most EMAS trials converged by that point. The standard error over these 50 trials is also provided in the table as a measure of the uncertainty of the data (see Sec. 3 for more detailed discussion of the significance of differences between groups). The performance of the evolving team, regardless of cooperation and fitness function used, is still superior to that of the individual algorithms used. For instance, the average performances of 100 trials of the individual algorithms executed on the sinusoidal function are -116.6 ± 8.1 for pattern search, -158.7 ± 5.3 for a genetic algorithm, -135 ± 8.4 for simulated annealing, and 42.35 ± 10.5 for Powell's method, compared to the worst value in Table 2 for the sinusoidal function of -165 ± 5. For a fixed fitness function, full cooperation produces better solutions than either partial or no cooperation for each numerical optimization problem. Furthermore, the solutions generated using partial cooperation are either better than or equivalent to (not significantly different than) solutions produced using no cooperation. It is also found that for all numerical optimization problems, the individual and assistant fitness functions are not significantly different from one another. The team fitness function; however, generally produces worse



(a) (b)

Fig. 4 Sample packings for the eight-cube problem (left) and for the arbitrary components problem (right). Intersections are shown in red/light gray and protrusions are shown in blue/dark gray.

solutions than the individual and assistant fitness functions. The only exceptions occur when there is no cooperation, for instance in the sinusoidal and fractal functions.

In evolutionary terms, team-based reward structures reduce the relative fitness of the best individuals, making it more difficult for them to stand out in the population. When the specific accomplishments of direct contributors or assistants are explicitly recognized in the fitness function, the likelihood that these agents will prosper and distinguish themselves from others is higher. As a result, it is expected that the performance of the organization as a whole will also improve. To support this claim, consider Fig. 5, in which the average and standard deviation of the percentage of agents running each algorithm is provided as a function of iteration for each fitness function. The data in Fig. 5 correspond to the results for EMAS applied using partial cooperation for the sinusoidal numerical optimization problem. It is observed that under the team fitness function, the population is evenly distributed among the available algorithms throughout the entire optimization process. By contrast, the individual and assistant fitness functions cause a dramatic divergence in the number of different agent types. For example, in the assistant fitness function, the number of pattern search agents is significantly higher than any of the other agent types. Similar trends are observed for all levels of cooperation and each of the other numerical optimization problems considered here.

4.2 Three-Dimensional Packing. The final objective function values for the eight-cube packing problem are provided in Table 3. Each data point corresponds to an average over five independent trials. The standard error is also provided as a measure of uncertainty in the average (see Sec. 3 for a more detailed discussion of the significance of differences across groups). For the individual and assistant fitness functions, full cooperation is better than no cooperation, but is not significantly different than the no cooperation result for the team fitness function. Regardless of fitness function, partial cooperation is either better than or equivalent to no cooperation and either worse than or equivalent to full cooperation. In addition, regardless of cooperation, the individual fitness function is better than the team fitness function. The assistant fitness function is either worse than or equivalent to the individual fitness function and is either better than or equivalent to the team fitness function. These results support those of numerical optimization.

For the arbitrary geometries, ten trials were run on just two of the groups: no cooperation + team fitness function and full cooperation + individual fitness function. These represent the extremes of the groups in terms of objective function value based on the previous observations. Because of the computation time associated with gathering data for all groups, it was considered sufficient to simply test that the trends hold for the more complex arbitrary geometry problems as well. The no cooperation + team

Table 2 Numerical optimization results (mean ± standard error) for 50 trials of EMAS

		Cooperation		
		None	Partial	Full
Bohachevsky function				
Fitness function	Individual	0.15 ± 0.02	0.10 ± 0.01	0.02 ± 0.01
	Assist	0.16 ± 0.02	0.10 ± 0.02	0.02 ± 0.01
	Team	0.26 ± 0.03	0.21 ± 0.02	0.10 ± 0.02
Sinusoidal function				
Fitness function	Individual	-172 ± 3	-174 ± 3	-193 ± 1
	Assist	-169 ± 3	-176 ± 3	-193 ± 1
	Team	-166 ± 4	-165 ± 5	-187 ± 2
Peaks function				
Fitness function	Individual	-449 ± 14	-510 ± 8	-632 ± 7
	Assist	-470 ± 14	-497 ± 9	-639 ± 6
	Team	-418 ± 14	-465 ± 11	-588 ± 11
Fractal function				
Fitness function	Individual	0.09 ± 0.01	0.07 ± 0.004	0.01 ± 0.003
	Assist	0.09 ± 0.01	0.07 ± 0.004	0.01 ± 0.003
	Team	0.09 ± 0.01	0.10 ± 0.01	0.06 ± 0.01

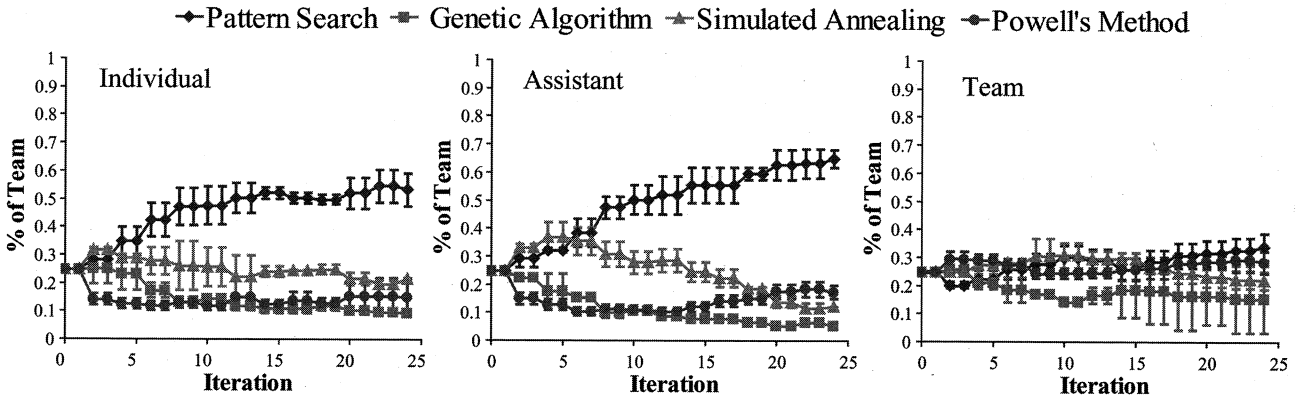


Fig. 5 Average and standard deviation of percentage of team running each algorithm over the course of 25 iterations of EMAS. Data presented for sinusoidal function under partial cooperation.

fitness function, with a mean of 0.0046 and standard error of 0.0014, is significantly worse than the full cooperation + individual fitness function, with a mean of 0.0013 and standard deviation of 0.0001 (see Sec. 3 for further discussion of significance of differences). This result supports those of the eight-cube packing problem and numerical optimization problems.

4.3 Significance of Differences Between Groups. Because of the lower limit on the objective function value (the global minimum), the data for both numerical optimization and 3D packing tended to be positively skewed rather than normally distributed. Furthermore, in the eight-cube packing problem, relatively few trials (five for each group) were conducted because of the computational overhead associated with each trial. These facts bring into question the appropriateness of using the mean and standard deviation as indicators of significant differences between the groups. In this section, this issue is addressed using more advanced statistical analysis tools [40]. For all statistical tests employed in this section, the significance level, α, was chosen as 0.05.

For numerical optimization, Levene's test for equality of variance was significant—meaning that the variance between groups was significantly different. Furthermore, normality tests on each group using the Kolmogorov–Smirnov (KS) test found that none of the groups displayed normal distributions. To compare groups, the KS test was used again. After application of the KS test, it is found that for each level of cooperation, both individual and assistant fitness functions are significantly better than the team fitness function, but are not significantly different from each other. Furthermore, it is found that for each fitness function, full cooperation is significantly better than partial and no cooperation, but partial and no cooperation are not significantly different from each other. These findings take into account error that may exist in making multiple comparisons within a common data set (type I error). The findings of these more rigorous nonparametric tests thus support the trends shown by the mean and standard error statistics in Table 2.

Table 3 Eight-cube packing results (mean ± standard error) for five trials of EMAS

		Cooperation					
		None		Partial		Full	
		Mean	St. Err.	Mean	St. Err.	Mean	St. Err.
Fitness function	Individual	0.463	0.003	0.345	0.049	0.259	0.053
	Assist	0.477	0.034	0.453	0.024	0.345	0.064
	Team	0.513	0.024	0.470	0.026	0.496	0.060

Because of the consistency of the numerical optimization data across groups (trends existed across levels of cooperation and across fitness functions), it can be assumed that there are no interaction effects between cooperation and fitness function. This means that each variable can be examined independently, so rather than comparing each of the fitness functions for each level of cooperation, the effect of a fitness function can be evaluated by looking at the data over all levels of cooperation. While there are only 5 data points for each group in the eight-cube packing problem, such an additive comparison means that groups of 15 data points are being compared, rather than groups of 5 data points. For the eight-cube packing problem, the data for each variable level was non-normal, though application of Levene's test found that the error variances were approximately equal across groups. Once again, the KS test was used to compare variable levels to one another. The KS test finds that full cooperation is significantly better than no cooperation, but partial cooperation is not significantly different from either. Likewise, it is found that the individual fitness function is significantly better than the team fitness function, but the assistant fitness function is not significantly different from either. This, again, tells the same story as is told by the mean and standard error statistics shown in Table 3.

As for numerical optimization and eight-cube packing, the data for the arbitrary geometries packing was not normally distributed. A KS comparison of the two groups of ten data points showed, however, that they were significantly different from each other, which supports the conclusions made earlier based on the mean and standard error.

5 Summary and Implications for Design

The observation that increasing cooperation improves the final objective function value reached by EMAS supports the use of cooperation in genetic approaches to engineering design. The data suggest that rather than evolving algorithms and strategies independently of one another, better solutions can be found by evolving strategies in a highly cooperative setting. Studies of nonevolving teams have long indicated that performance of independent agents may be enhanced by requiring cooperation and/or collaboration between them [10,17]. However, it was not explicitly shown in previous work on EMAS that such cooperation *among evolving entities* also boosts performance. The results of this study indicate that cooperation in evolving teams is in fact beneficial. From this it is recommended that in cases where the best strategies and parameters are not clear, strategies should be allowed to cooperate and evolve while working toward a solution. While evolution implies that strategies adapt themselves to the problem space, cooperation ensures that useful knowledge of the space (in the

form of newly generated solutions) is communicated between strategies and across generations. The difference in performance between fitness functions supports both computational and human organizational studies on team behavior under different reward scenarios [15,16,41–45]. For example, Wageman and Baker [15] observed that in human teams, high reward interdependence (i.e., team fitness functions) when task interdependence is low (each individual does not require the help of others) decreases overall performance. Despite the similarity in cooperation and reward interdependence trends between EMAS and organizational studies, it is important to note that the algorithm presented in its current form is not intended to model realistic design team scenarios.

Acknowledgment

This work was funded by NSF Grant No. BCS0717957. The authors would also like to acknowledge helpful input from and discussions with Matthew Wood (Carnegie Mellon University).

References

[1] Jacobson, S. H., and Yucesan, E., 2004, "Analyzing the Performance of Generalized Hill Climbing Algorithms," J. Heuristics, **10**, pp. 387–405.
[2] Cagan, J., Shimada, K., and Yin, S., 2002, "A Survey of Computational Approaches to Three-dimensional Layout Problems," Comput. Aided Des., **34**(8), pp. 597–611.
[3] Smith, J., and Fogarty, T. C., 1996, "Self-Adaptation of Mutation Rates in a Steady State Genetic Algorithm," International Conference on Evolutionary Computation, pp. 318–323.
[4] Moral, R. J., and Dulikravich, G. S., 2008, "Multi-Objective Hybrid Evolutionary Optimization with Automatic Switching Among Constituent Algorithms," AIAA J., **46**(3), pp. 673–681.
[5] Gomes, C. P., and Selman, B., 2001, "Algorithm Portfolios," Artif. Intell., **126**, pp. 43–62.
[6] Rachlin, J., Goodwin, R., Murthy, S., Akkiraju, R., Wu, F., Kumaran, S., and Das, R., 1999, "A-Teams: An Agent Architecture for Optimization and Decision-Support," Lect. Notes Comput. Sci., **1555**, pp. 261–276.
[7] DeSouza, P. S., and Talukdar, S. N., 1993, "Asynchronous Organizations for Multi-Algorithm Problems," Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice, Indianapolis, IN, USA, Feb. 14–16, pp. 286–293.
[8] Russell, S. J., and Norvig, P., 1995, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ.
[9] Talukdar, S. N., Baerentzen, L., Gove, A., and DeSouza, P. S., 1998, "Asynchronous Teams: Cooperation Schemes for Autonomous Agents," J. Heuristics, **4**, pp. 295–321.
[10] Campbell, M. I., Cagan, J., and Kotovsky, K., 1999, "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment," Res. Eng. Des., **11**(3), pp. 172–192.
[11] Nii, H. Y., 1986, "Blackboard Systems," Stanford University, Technical Report No. STAN-CS-86-1123/KSL-86-18.
[12] Hanna, L., and Cagan, J., 2009, "Evolutionary Multi-Agent Systems: An Adaptive and Dynamic Approach to Optimization," J. Mech. Des., **131**(1), pp. 011010-1–011010-8.
[13] Hanna, L., and Cagan, J., 2008, "Evolutionary Multi-Agent Systems, An Adaptive Approach to Optimization in Dynamic Environments," Proceedings of the ASME International Design Engineering Technical Conference and Comp. and Information in Eng. Conference IDETC/CIE, Design Automation Conference, New York, NY, Aug. 3–6.
[14] Potter, M. A., and DeJong, K. A., 2000, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," Evol. Comput., **8**(1), pp. 1–29.
[15] Wageman, R., and Baker, G., 1997, "Incentives and Cooperation: The Joint Effects of Task and Reward Interdependence on Group Performance," J. Organiz. Behav., **18**, pp. 139–158.
[16] Lawler, E. E., 2000, Rewarding Excellence: Pay Strategies for the New Economy, Jossey-Bass, San Francisco.
[17] Olson, J. T., and Cagan, J., 2004, "Inter Agent Ties in Team-Based Computational Configuration Design," Artif. Intell. Eng. Des. Anal. Manuf., **18**(2), pp. 135–152.
[18] Torczon, V., and Trosset, M. W., 1998, "From Evolutionary Operation to Parallel Direct Search: Pattern Search Algorithms for Numerical Optimization," Proceedings of the 29th Symposium on the Interface, Houston, TX, May 14–17, David W. Scott, ed., Comput. Sci. Stat., **29**(1), pp. 396–401.
[19] Yin, S., and Cagan, J., 2000, "An Extended Pattern Search Algorithm for Three-Dimensional Component Layout," J. Mech. Des., **122**(1), pp. 102–108.
[20] Aladahalli, C., Cagan, J., and Shimada, K., 2007, "Objective Function Based Pattern Search–An Implementation for 3D Component Layout," J. Mech. Des., **129**, pp. 255–265.
[21] Haupt, R. L., and Haupt, S. E., 2004, *Practical Genetic Algorithms*, Wiley, Hoboken, NJ.
[22] Ikonen, I., Biles, W. E., Kumar, A., Regade, R. K., and Wissel, J. C., 1997, "A Genetic Algorithm for Packing Three-Dimensional Non-Convex Objects Having Cavities and Holes," Proceedings of the Seventh International Conference on Genetic Algorithms.
[23] House, R. L., and Dagli, C. H., 1992, "An Approach to Three-Dimensional Packing Using Genetic Algorithms," Intell. Eng. Syst. Through Artif. Neural Networks, **2**, pp. 937–942.
[24] 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.
[25] Huang, M. D., Romeo, F., and Sangiovanni-Vincentelli, A., 1986, "An Efficient General Cooling Schedule for Simulated Annealing," IEEE International Conference on Computer Aided Design–Digest of Technical Papers, pp. 381–384.
[26] Szykman, S., and Cagan, J., 1995, "A Simulated Annealing-Based Approach to Three-Dimensional Component Packing," J. Mech. Des., **117**(2A), pp. 308–315.
[27] Cagan, J., Degentesh, D., and Yin, S., 1998, "A Simulated Annealing-Based Algorithm Using Hierarchical Models for General Three-Dimensional Component Layout," Comput. Aided Des., **30**(10), pp. 781–790.
[28] Kolli, A., Cagan, J., and Rutenbar, R. A., 1996, "Packing of Generic, Three-Dimensional Components Based on Multi-Resolution Modeling," Proceedings of the 22nd ASME Design Automation Conference (DAC-1479), Irvine, CA.
[29] Chapra, S. C., and Canale, R. P., 2002, *Numerical Methods for Engineers*, McGraw-Hill, New York.
[30] Fadel, G. M., Sinha, A., and McKee, T., 2001, "Packing Optimization Using a Rubberband Analogy," Proceedings of the 2001 ASME Design Engineering Technical Conferences and Computer Information Engineering Conference, Pittsburgh, PA.
[31] Tiwari, S., Fadel, G. M., and Fenyes, P., 2008, "Fast and Efficient Compact Packing Algorithm for Free-Form Objects," Proceedings of the 34th ASME Design Automation Conference, New York.
[32] Aladahalli, C., Cagan, J., and Shimada, K., 2007, "Objective Function Effect-Based Pattern Search: Theoretical Framework Inspired by 3D Component Layout," J. Mech. Des., **129**(3), pp. 243–254.
[33] Fournier, A., Fussell, D., and Carpenter, L., 1982, "Computer Rendering of Stochastic Models," Commun. ACM, **25**(6), pp. 371–384.
[34] Miller, G. S., 1986, "The Definition and Rendering of Terrain Maps," International Conference on Computer Graphics and Interactive Techniques, pp. 39–48.
[35] Sorkin, G. B., 1992, *Theory and Practice of Simulated Annealing on Special Energy Landscapes*, University of California Press at Berkeley, Berkeley, CA.
[36] Dowsland, K. A., and Dowsland, W. B., 1992, "Packing Problems," Eur. J. Oper. Res., **56**, pp. 2–14.
[37] Martello, S., Pisinger, D., and Vigo, D., 2000, "The Three-Dimensional Bin-Packing Problem," Oper. Res., **48**(2), pp. 256–267.
[38] Grignon, P. M., and Fadel, G. M., 2004, "A GA Based Configuration Design Optimization Method," J. Mech. Des., **126**(6), pp. 6–16.
[39] Mortensen, M. E., 1997, *Geometric Modeling*, Wiley, New York.
[40] Field, A., 2005, *Discovering Statistics Using SPSS*, Sage, London.
[41] Hackman, J. R., 1987, "The Design of Work Teams," in *Handbook of Organizational Behavior*, J. W. Lorsch, ed., Prentice-Hall, Englewood Cliffs, NJ, pp. 315–342.
[42] Carley, K. M., and Prietula, M. J., eds., 1994, *Computational Organization Theory*, Erlbaum, Hillsdale, NJ.
[43] Sarin, S., and Mahajan, V., 2001, "The Effect of Reward Structures on the Performance of Cross-Functional Product Development Teams," J. Marketing, **65**, pp. 35–53.
[44] Wilkins, D. E., and Lawhead, P. B., 2000, "Evaluating Individuals in Team Projects," ACM Special Interest Group Comput. Sci. Educ. Bull., **32**(1), pp. 172–175.
[45] Hayes, J. H., Lethbridge, T. C., and Port, D., 2003, "Evaluating Individual Contribution Toward Group Software Engineering Projects," Proceedings of the 25th International Conference on Software Engineering, pp. 622–627.