Protocol-Based Multi-Agent Systems: Examining the Effect of Diversity, Dynamism, and Cooperation in Heuristic Optimization Approaches

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

Jonathan Cagan
e-mail: cagan@cmu.edu

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

Many heuristic optimization approaches have been developed to combat the ever-increasing complexity of engineering problems. In general, these approaches can be classified based on the diversity of the search strategies used, the amount of change in these search strategies during the optimization process, and the level of cooperation between these strategies. A review of the literature indicates that approaches that are simultaneously very diverse, highly dynamic, and cooperative are rare but have immense potential for finding high quality final solutions. In this work, a taxonomy of heuristic optimization approaches is introduced and used to motivate a new approach called protocol-based multi-agent systems. This approach is found to produce final solutions of much higher quality when its implementation includes the use of multiple search protocols, the adaptation of these protocols during the optimization, and the cooperation between these protocols than when these characteristics are absent.
[DOI: 10.1115/1.4003290]

1 Introduction

The approaches that have been developed to solve complex optimization problems are numerous and diverse. From evolutionary computing to agent-based optimization, these approaches vary not only in structure but also in efficacy and efficiency. It may not always be clear which approach and corresponding parameter settings are best for a particular problem. This is especially the case when the characteristics of the problem space are not well known. Practitioners therefore face several challenges when deciding on an approach to use on a new optimization problem.

In many cases, an approach is simply an *algorithm*, an iterative procedure consisting of the sequential application of search *strategies*. Here, a strategy is defined as a method for searching the solution space that consists of a method for generating a set of solutions from an input set of solutions (the *generating protocol*) and a method for selecting one or more of those generated solutions (the *selecting protocol*). An approach can also be defined as the implementation of many algorithms in parallel (as in algorithm portfolios) or as the application of multiple cooperating algorithms (as in asynchronous teams of agents). While approaches vary significantly, there are several characteristics that are used here to classify the broad array of heuristic (i.e., nondeterministic and nonclassical) literature by breaking the approaches into their constituent strategies and protocols. These characteristics are (i) the diversity of the strategies available at any time in the approach, (ii) the amount of change that occurs with respect to the strategies during the optimization, and (iii) the extent to which there is "cooperation" between strategies.

The first element in the taxonomy is the diversity of the strategies available in an approach. If at every point in the optimization only a single strategy is available to be employed, the approach is *homogeneous*. On the other hand, if there is more than one strategy available, the approach is *heterogeneous*. The availability of more than one strategy does not necessarily mean that more than one will actually be applied. For example, in simulated annealing, multiple neighborhoods are available at each iteration, making the approach heterogeneous, even though only one neighborhood is actually explored in that iteration.

The second element in the taxonomy is the amount that strategies change based on their performance over the course of the algorithm. If change does not occur or is specified completely a priori, the approach is *static*. If the resources allotted to strategies (likelihood of application, CPU time, etc.) change dynamically, the approach is *resource dynamic*. On the other hand, if the actual structure of strategies is adjusted dynamically, then the approach is *structure dynamic*. Finally, if both the structure of and the allocation of resources to a strategy change dynamically, the approach is *fully dynamic*. It is important to mention here that diversity is independent of dynamism—if there is only a single strategy available at any time, even if that strategy changes over time, the approach is still homogeneous.

The final element of the taxonomy is the level of cooperation between strategies, which is measured by the number of individual strategies that may be applied on a single solution simultaneously (which indicates sharing of that solution). Most optimization approaches are *noncooperative*, meaning that only one strategy is applied to a solution at a time. In *partially cooperative* approaches, some solutions are shared at some times. Finally, *fully cooperative* approaches allow the sharing of all solutions between strategies at all times.

In Sec. 2, some of the representative optimization literature is presented with respect to these classifications. It will be demonstrated that there are relatively few approaches that are simultaneously heterogeneous, structure or fully dynamic, and partially or fully cooperative. This represents a research opportunity gap for approaches that employ multiple strategies that change based on the performance of the algorithm and share solutions among themselves. In later sections, such an approach, called protocol-based multi-agent systems (PMAS), is developed and is shown to produce higher quality solutions than approaches that are lacking in one of these areas.

Manuscript received February 21, 2010; final manuscript received December 8, 2010; published online January 24, 2011. Assoc. Editor: Yan Jin.

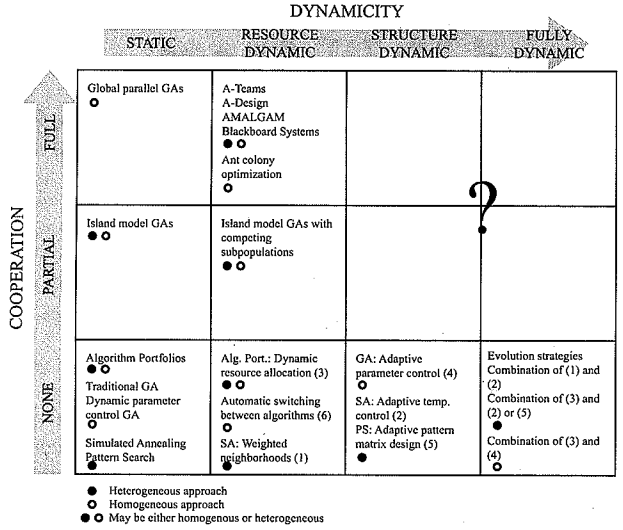

Fig. 1 Taxonomy for optimization approaches based on diversity, dynamism, and cooperation

2 Background

Figure 1 plots a wide range of heuristic optimization approaches with respect to the taxonomy developed here. These approaches will be discussed in more detail in this section, but it is apparent from the figure that there is a research opportunity gap for approaches that are simultaneously heterogeneous, structure and fully dynamic, and cooperative. Based on some of the findings in the literature, it appears that such approaches would also be likely to have better solution quality than the others as well.

2.1 Diversity of Strategies. Homogeneous optimization approaches include traditional genetic algorithms [1], harmony search [2], and other population-based approaches. Adaptive extensions to genetic algorithms, such as dynamic parameter control and adaptive parameter control, are likewise homogeneous [3]. Similarly, global and coarse-grained parallel genetic algorithms are homogeneous as well if they employ a single strategy for the entire optimization [4]. Algorithm portfolios in this category run many copies of the same algorithm in parallel, though the algorithms may be seeded with different random number generators or starting points [5]. In general, homogeneous approaches are only as effective as the single strategy or strategy progression used.

Heterogeneous optimization approaches include many traditional algorithms such as simulated annealing and pattern search [6,7]. In addition, algorithm portfolios in which different algorithms or heuristics are available at the same time are also heterogeneous [5,8–10]. Similarly, agent-based methods for optimization such as A-Teams [11,12] and A-Design [13] embody different strategies in independent agents that act simultaneously, making these approaches heterogeneous as well. Self-adaptive evolution strategies [14] and extensions to coarse-grained or island model genetic algorithms, such as those of Pohlheim [15], are also heterogeneous. Heterogeneous approaches can improve solution quality by decreasing the risk associated with relying on a single strategy at a time [16]. It is for this reason that it is speculated that heterogeneous approaches will have better solution quality than homogeneous approaches.

2.2 Dynamism. Static approaches include traditional genetic algorithms, simulated annealing, and pattern search. Genetic algorithms with predictably dynamic mutation rates, such as those outlined by Fogarty [17], Hesser and Manner [18,19], and Back and Schutz [20], are also static. Algorithm portfolios using these static algorithms that prespecify the design of the portfolio (which algorithms will be used) and provide constant resources to each algorithm are also static. Global and course-grained genetic algorithms are static when the same number of solutions (resources) are allotted to each strategy, and neither that number nor the strategies change. Though static approaches make the prediction of performance easier and require little if any bookkeeping during the optimization, there is no guarantee in static approaches that the strategies chosen are ideal.

Resource dynamic extensions exist for genetic algorithms, simulated annealing, and pattern search. For example, the use of feedback from the search to guide the probability of searching a priori set neighborhoods or patterns is resource dynamic [21,22]. Similarly, course-grained genetic algorithms with competition between subpopulations [15] alter the number of resources (solutions) allotted to each strategy based on the performance of each subpopulation employing that strategy without actually changing the strategies themselves. Dynamic resource allocation in algorithm portfolios [23,24] as well as automatic switching schemes for algorithm portfolios [25,26] are also resource dynamic. In addition, agent-based methods, such as A-Teams and A-Design, and ant colony optimization are resource dynamic as well [11,13,27]. Bookkeeping is generally more of a concern with resource dynamic approaches (as opposed to static approaches) and the question of whether or not the strategies chosen initially are in fact the most appropriate for the problem still remains in these approaches.

Structure dynamic approaches include further extensions to genetic algorithms, simulated annealing, and pattern search, as well as portfolios of these extended algorithms. Adaptive parameter control [3] is an alteration to traditional genetic algorithms, which makes them structure dynamic. Similarly, the adaptive temperature decrement extension to simulated annealing [28] adaptively sets the temperature for the system based on the statistics of the search. Finally, new patterns can be generated in pattern search based on previously successful patterns [21]. While structure dynamic approaches do not adjust resources based on strategy performance, they are more likely to uncover promising regions of the strategy space than either static or resource dynamic approaches. For many traditional algorithms, it has been shown that both resource and structure dynamic extensions individually can improve performance over static approaches [15,21,22,28,29].

Approaches in the resource dynamic and structure dynamic categories can be combined to create fully dynamic approaches. For instance, the adaptive temperature decrement and the updating of neighborhood probability extensions to simulated annealing can be combined to form a fully dynamic approach [30]. Dynamic resource allocation for algorithm portfolios containing structure dynamic algorithms is also fully dynamic. In addition to these hybrid approaches, though, self-adaptive evolution strategies are fully dynamic as their strategies are encoded in the chromosome of individual solutions, so both the structure and the prevalence of strategies evolve [14]. It has been observed that fully dynamic evolution strategies have performance advantages over static genetic algorithms [31]. This observation supports the hypothesis that more dynamic approaches have performance advantages over less dynamic approaches. With increasing dynamism, the amount of bookkeeping necessary to evaluate strategies increases, which in turn may increase the computation time of the approach.

2.3 Cooperation Between Strategies. Noncooperative approaches include the traditional heuristic algorithms that only apply a single strategy on a single solution at a time (simulated annealing and pattern search) as well as those that apply a single strategy to all solutions in a time step (genetic algorithms). Algorithms in a portfolio typically do not share solutions, making algorithm portfolios noncooperative as well. Even the adaptive and automatic switching approaches to algorithm portfolios are noncooperative because of this lack of sharing. Coarse-grained, or island model, genetic algorithms, which do not have migration (sharing of solutions) between populations, are very similar to algorithm portfolios in this way. In evolution strategies, each in-

dividual solution determines the single strategy that will be applied to it, and no more than one strategy will be applied to that solution at the same time.

Commonly used partially cooperative approaches include island model genetic algorithms that employ migration between subpopulations. The frequency, schedule, and mode of migration can affect both the final solution quality and the speed of convergence [32].

Fully cooperative approaches include multi-agent and multi-algorithm approaches, such as A-Teams [33], A-Design [13], and AMALGAM [34], in which strategic agents share solutions fully. Blackboard systems [35] are also fully cooperative, as many agents may utilize the same solution information simultaneously to form new solutions. Synergy, the ability of a group to outperform its individuals acting independently, is an emergent behavior in the agent-based fully cooperative optimization approaches [12]. This suggests that approaches that are cooperative have performance advantages over noncooperative approaches. However, more cooperation often implies more communication, either between agents or between agents and a common database. As a result, cooperative approaches can often mean higher computational overhead.

2.4 Classification of Generating Versus Selecting Protocols. Generating and selecting protocols may have different classifications in an approach. For instance, several dynamic evolutionary approaches focus on altering only the generating protocol (e.g., the mutation rate) and do not change the selecting protocol [3,15]. Non-evolutionary approaches such as Huang's [28] extension to simulated annealing have a generating protocol classified as heterogeneous, static, and noncooperative and a selecting protocol classified as homogeneous, structure dynamic, and noncooperative. The fact that approaches may be classified at the level of their generating and selecting protocols represents a unique opportunity to study the effect of diversity, dynamism, and cooperation at a finer resolution.

2.5 Evolutionary Multi-Agent Systems. In earlier work, Hanna and Cagan [36,37] developed a heterogeneous, structure dynamic, and fully cooperative optimization approach called Evolutionary Multi-Agent Systems (EMAS) in which evolving agents embodied independent algorithms and cooperated to solve complex optimization problems. EMAS was found to produce high quality solutions when applied to traveling salesman problems, numerical optimization problems, and 3D packing problems. One challenge associated with implementing EMAS, however, is its large computational expense, which arises from its having to run many complete algorithms sequentially. While EMAS evolves the agents to take advantage of algorithms that are most successful at each point in the optimization process, it cannot distinguish which pieces of those algorithms are the source of its strength. Therefore, the agent population in EMAS may not be as effective as it could be because effective search procedures (generating and selecting protocols) may be overlooked if others in the algorithm are not performing well, or ineffectual procedures may be run more often if others in the algorithm make up for their shortcomings. This drawback will be addressed in this work as well.

3 Protocol-Based Multi-Agent Systems

A new approach based on EMAS, but which breaks down algorithms into their generating and selecting protocols, is presented here. This approach, called PMAS, consists of the parallel implementation of generating and selecting protocols, each represented by independent agents. When activated, *generating agents* apply their generating protocol to the solutions available to them and then pass those solutions to the selecting agents. *Selecting agents*, when activated, select and then pass solutions back to the generating agents. Each type of agent works in parallel with all other agents of that type. A single "iteration" of the PMAS algorithm is defined as the parallel activation of all generating agents followed by the parallel activation of all selecting agents.

The PMAS framework provides a platform for studying the effect of diversity, dynamism, and cooperation at the level of search protocols. The generating and selecting protocols used (see Sec. 5) represent a kind of "primordial soup" of algorithm building blocks, from which traditional genetic algorithms, simulated annealing, and pattern search can all be built. Using these building blocks, the PMAS platform can mimic incredibly diverse behaviors in optimization—from the parallel implementation of many of these traditional algorithms as in algorithm portfolios (heterogeneous, static, and noncooperative implementation) to a cooperative evolving team of agents employing them as in EMAS (heterogeneous, structure dynamic, and fully cooperative implementation)—simply by altering the characteristics of the agents and their communication structure. For example, the heterogeneity of PMAS will be altered by changing how many different agent types there are. The dynamism can be altered by allowing agents to evolve over time. Finally, different levels of cooperation can be implemented by allowing more or less sharing of solutions. In Sec. 4, each of the characteristics of PMAS that can be changed to address different behaviors will be explained in detail. Based on the findings of EMAS, its extensions, and other literature, it is hypothesized that the more diverse, dynamic, and cooperative implementations of PMAS will have higher solution quality. This hypothesis will be tested here by implementing each classification in the taxonomy and examining its effect on solution quality and computation time. This will allow for the comparison of many different optimization approaches using a single architecture.

The different PMAS implementations will be applied to solve both the one-max problem and the problem of packing the 13 arbitrary 3D geometries given in Aladahalli et al. into a cubical container [38]. The one-max problem is a well known test problem for genetic algorithms in which the goal is to maximize the number of ones in a binary string of length n [39]—or, equivalently, to minimize the string's hamming distance (number of incorrect bits). Though evaluation of the hamming distance is trivial, there are 2^n possible strings, which means that the solution space can be extremely large. Despite its simplicity, though, the one-max problem has few direct applications in engineering, which is why the 3D packing problem will also be explored.

Solutions for the 3D packing problem are represented here by lists of real numbers corresponding to positions and orientations for components. The number of variables for a 3D packing solution is six times the number of components since each component is described by its x, y, and z coordinates and the rotations about its x, y, and z axes. The components and container in the 3D packing problem are represented by octrees, which are discretizations of the component/container geometries into cuboidal volumes called voxels [40]. The objective function for the 3D packing problem can be written as

$$g(\vec{x}) = \sum_{i=1}^{C-1} \sum_{j=1}^{C} I[\theta_i(\vec{x}), \theta_j(\vec{x})] + \sum_{k=1}^{C} P[\theta_k(\vec{x})]$$

where $\vec{x}$ is a vector of the positions and orientations of all C components, $\theta_c(\vec{x})$ represents the octree for component c with the location given in $\vec{x}$, $I[\theta_i(\vec{x}), \theta_j(\vec{x})]$ is the volume of voxels in $\theta_i(\vec{x})$ that intersect voxels in $\theta_j(\vec{x})$, and $P[\theta_k(\vec{x})]$ is given by

$$P[\theta_k(\vec{x})] = \sum_{v} [V_v \times d(v,\vec{x}) \times \delta_v]$$

which is a summation over all voxels making up $\theta_k(\vec{x})$ where V_v is the volume of voxel v and $d(v,\vec{x})$ is the distance between the center of voxel v in $\theta_k(\vec{x})$ and the center of the container normalized by the container side length. The function δ_v is similar to a

Table 1 Protocols and parameters used for PMAS implementation

			Parameter (one-max and 3D packing)	Values				Parameter (3D packing only)	Values			
				00	01	10	11		00	01	10	11
Generating protocols	00	Crossover	Probability of application	0.99	0.9	0.85	0.75	α	0	0.1	0.25	0.5
	01	Meiosis	–					Excess	0	0.1	0.25	0.5
	10	Mutation	Probability of application	0.001	0.01	0.1	0.15	σ	0.01	0.1	0.25	0.5
	11	Local search	Maximum distance (number of variables)	0.001	0.01	0.1	0.2	Step size	0.001	0.01	0.1	0.25
Selecting protocols	00	Roulette wheel	Selection pressure	1.25	1.75	2.25	2.75	–				
	01	Rank	–					–				
	10	Tournament	Probability that worse contender wins	0.01	0.05	0.1	0.25	–				
	11	Elitist	–					–				

Kronecker delta function in that $\delta_v=1$ when v protrudes and $\delta_v=0$ otherwise.

The large number of variables, the fact that variables may take on any value, and the multimodal nature of the space make 3D packing a much more computationally demanding optimization problem than one-max.

4 Algorithm Description

4.1 Structure of PMAS Agents. Both generating and selecting agents in PMAS are represented by a 4 or 6 bit binary chromosome. This chromosomal representation allows for the implementation of structure and full dynamism in the form of evolution of the agents (see Sec. 5). The first 2 bits encode the method gene, which determines which of four possible generating or selecting protocols the agent will use. The last 2 to 4 bits encode the parameter genes, which identify the values of the parameters associated with the method given by the method gene. For the one-max problem, there are 2 bits encoding the parameter gene, and the overall chromosome length for agents is 4. For the 3D packing problem, there are up to two parameters associated with each protocol, so the chromosome has a length of 6: one 2 bit method gene and two 2 bit parameter genes. The protocols used, their associated parameters, and the corresponding gene values are given in Table 1 and will be discussed further in this section. The protocols listed in Table 1 are common in the optimization literature, and the parameter values used are considered appropriate based on the same literature. Additional protocols and parameter values may be incorporated simply by altering the gene representation of the agents. Though an in-depth analysis of the effect of utilizing different protocols and parameter values is beyond the scope of this study, future analysis of PMAS might include a study of how different protocols need to be for diversity to have an effect on performance.

4.2 Generating Protocols in PMAS. The generating protocols used here are crossover, meiosis, mutation, and local search, corresponding to generating agent method gene values of 00, 01, 10, and 11, respectively. The parameters for crossover are the probability of application and the α parameter for performing arithmetic crossover (3D packing only). For meiosis, the only parameter is the amount of "excess" on either side of the parents' range of values (3D packing only). The parameters for mutation are the probability of application and the σ parameter for Gaussian mutation (3D packing only). Finally, the parameters for local search are the maximum number of variables to be changed and the size of the range of possible new values for those variables (3D packing only).

4.3 Selecting Protocols in PMAS. The selecting protocols used here are roulette wheel selection, rank selection, tournament selection, and elitist selection, corresponding to selecting agent method gene values of 00, 01, 10, and 11, respectively. The parameter associated with roulette wheel selection is the selection pressure, which is used for both the one-max and 3D packing implementations. For rank selection, there are no associated parameters for either the one-max problem or the 3D packing problem implementations. The parameter associated with tournament selection for both the one-max and 3D packing implementations is the likelihood of the worse competitor winning the tournament. There are no parameters associated with elitist selection for either the one-max or the 3D packing implementations.

5 Methodology

5.1 Implementation of Diversity in PMAS. Homogeneity is implemented in PMAS by making all generating agents identical to one another and all selecting agents identical to one another every iteration. Heterogeneity is implemented by randomly selecting different protocols for each agent.

5.2 Implementation of Dynamism in PMAS. Dynamism is controlled by adjusting agents' definitions and/or the number of solutions they may act upon. The implementations for the four levels of dynamism are as follows.

5.2.1 Static Implementation. The static implementation keeps each agent's protocol constant over the course of the algorithm. Furthermore, agents always act upon (generate or select) the same number of solutions every iteration.

5.2.2 Resource Dynamic Implementation. In the resource dynamic implementation, agents' protocols do not change over the course of the algorithm. However, the number of solutions an agent acts upon changes every iteration based on its rank in the population, which is determined by the relative fitness, f, of the agent. For each generating agent, fitness is defined as

$$f_g=\frac{s}{g} \tag{1}$$

where g is the number of solutions generated by the agent and s is the number of those solutions that were then selected by the selecting agents, which must be less than or equal to g. Equation (1) specifies that each generating agent has a fitness between 0 and 1. For each selecting agent, fitness is defined as

$$f_s = \frac{\min(b,u)}{s} \tag{2}$$

where b is the number of solutions that the agent selected that were better than or equal to the median of all solutions selected during the same iteration by all selecting agents, u is the number of unique solutions selected by the agent, and s is the total number of solutions the agent has selected, which must necessarily be greater than or equal to b and u. The uniqueness count is used to prevent early convergence. For selecting agents, fitness is in the range $0 \leq f_s \leq 1$.

Given its fitness, an agent is ranked from 0 to -1, where P is the number of generating agents or selecting agents. The number of solutions an agent may act upon (its ration) is adjusted after each iteration such that

$$r_{i+1}(a) = r_{\min} + \frac{2R(a)[r_i(a) - r_{\min}]}{P-1} \tag{3}$$

where $r_i(a)$ is the ration for agent a at iteration i, $r_{\min}$ is the minimum ration that may be allotted to any agent, and $R(a)$ is the rank of agent a, where 0 represents the worst agent and $P-1$ represents the best agent. Equation (3) specifies that the middle agent receives the same ration as the previous iteration, the worst agent receives the minimum ration, and the best agent receives twice the number of solutions above the minimum ration as it had the previous iteration. The total number of solutions allotted to the population of agents is held constant, so if this total is disrupted by the rationing process, randomly chosen agents' rations are increased or decreased until the total is reached. Here, the minimum ration is 10 for the one-max problem and 5 for the 3D packing problem. The initial ration for agent is 50 in the one-max problem and 25 for the 3D packing problem. The minimum and initial rations for agents were chosen somewhat arbitrarily but were lower for the 3D packing problem due to the computational expense of evaluating the objective function. For the same population size, a higher ratio of minimum ration to initial ration levels the playing field for poorly performing agents, which could prevent premature convergence but could also act to slow the progress of an otherwise successful team. A detailed analysis of the effect of minimum and initial rations on computation time and solution quality is a subject of future study.

5.2.3 Structure Dynamic Implementation. Structure dynamism is implemented by using harmony search to replace the agent with the worst fitness with a new agent whose chromosome is constructed by randomly choosing alleles from the other agents in the population. In this way, the agent team is allowed to adapt during the optimization. There is a small probability of mutation in each allele of the resulting agent's chromosome. Here, this probability is 0.01 per allele.

5.2.4 Fully Dynamic Implementation. To implement fully dynamic PMAS, both the readjustment of agent rations using Eq. (3) and harmony search described above are used.

5.3 Implementation of Cooperation in PMAS. In order for agents to cooperate fully, they must be able to act on all solutions created by other agents simultaneously. On the other hand, noncooperation means that each solution is only acted upon by one agent—here, the agent that acted upon it previously. Partial cooperation involves the occasional sharing of some solutions. The implementation for each of these levels of cooperation is discussed below.

5.3.1 Noncooperative Implementation. A noncooperative scenario for PMAS requires that each selecting agent keep track of the generating agents that created the solutions it selected and vice versa for the generating agents. To implement this, a solution is "tagged" by the last agent to act upon it. Agents have "inboxes" where solutions may be placed, so generating agents place solutions that have been tagged by a particular selecting agent in that selecting agent's inbox, and vice versa. Noncooperative agents may only act upon solutions that are in their inboxes and do not share these solutions with any other agents. If there are no solutions in an agent's inbox (none of its solutions were subsequently acted upon), the agent uses the solutions that were in its inbox the previous iteration. At the start of the algorithm, when solutions are randomly assigned to generating agents, there are no selecting agent tags, so generating agents place solutions in randomly chosen selecting agents' inboxes.

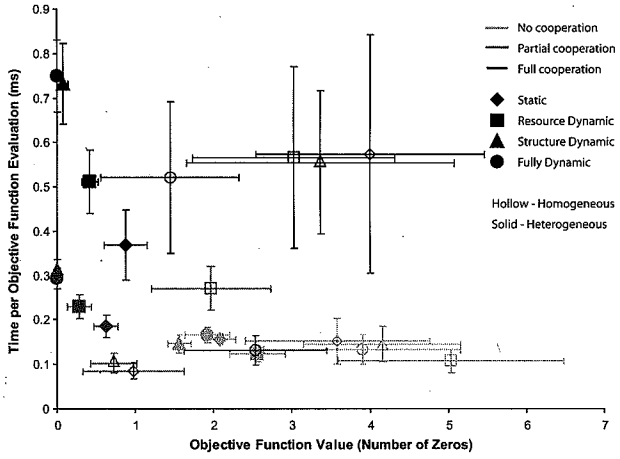

Fig. 2 Average time per objective function evaluation (ms) versus final average objective function value for different optimization classifications using the PMAS platform with identical agent population sizes and number of iterations executed on the one-max problem

5.3.2 Partially Cooperative Implementation. As in the noncooperative implementation, agents in the partially cooperative implementation may only act upon solutions in their inbox and pass solutions to the last agents that tagged those solutions. In the partially cooperative implementation, however, the best solutions generated are shared among selecting agents and the best solutions selected are shared among generating agents. This broadcasting idea is similar to the migration protocol used by Pohlheim [15] for island model genetic algorithms.

5.3.3 Fully Cooperative Implementation. Agents in fully cooperative PMAS do not send solutions to particular inboxes but instead post all solutions to a common "blackboard." This blackboard only holds solutions from a single generating or selecting procedure at a time and is erased after all generating or selecting agents have acted. Agents may act upon any solutions in the blackboard rather than only being allowed to act upon solutions in their inboxes.

An example of a single iteration of heterogeneous, fully dynamic, and fully cooperative PMAS is provided in the Appendix.

6 Results

6.1 One-Max Problem. The results of implementing each classification in PMAS on the one-max problem are shown in Fig. 2, which plots the final average objective function value against the average time per objective function evaluation when the number of agents and the number of iterations are prespecified. The time per objective function evaluation represents the "cost" associated with each objective function evaluation in a run (25 iterations) of PMAS and is calculated by dividing the total computation time of the run by the number of function evaluations in the run. By doing this, the computational overhead of different classifications can be compared for structurally identical PMAS runs

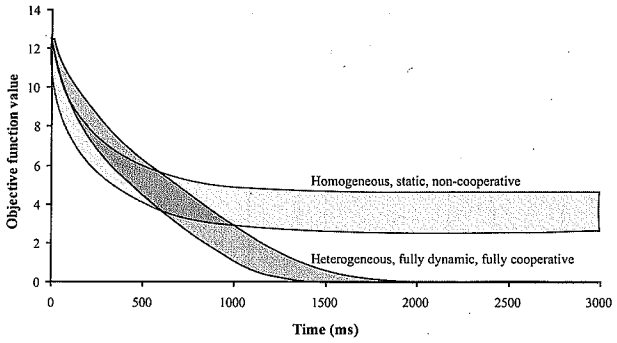

Fig. 3 Computation time versus mean and standard error range when both a heterogeneous, fully dynamic, and fully cooperative approach and a homogeneous, static, and noncooperative approach are implemented in PMAS and applied to the one-max problem

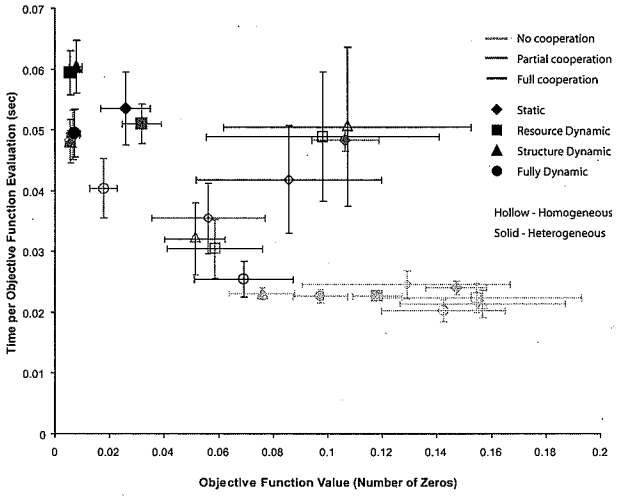

Fig. 4 Average time per objective function evaluation (s) versus final average objective function value for different optimization classifications using the PMAS platform with identical agent population sizes and number of iterations executed on the 3D packing problem

(i.e., where the number of agents and number of iterations are prespecified). The error bars shown in Fig. 2 represent the standard error of the mean over 50 trials.

As shown in Fig. 2, heterogeneity leads to better solution quality (lower objective function value) than homogeneity. Homogeneity also has a higher variance in solution quality, which may be due to the fact that the solution quality of a homogeneous run is highly dependent on the single protocol used by all agents.

The diversity and cooperation trends are less clear when the approach is homogeneous than when it is heterogeneous. In the heterogeneous case, for example, the structure and fully dynamic approaches produce better quality solutions than static and resource dynamic approaches. These trends are not as defined, however, in the homogeneous case.

Trends for solution quality are similarly muddled when comparing different levels of cooperation in the homogeneous case. The heterogeneous case, however, indicates that both partial and full cooperation are better than no cooperation but are not significantly different from one another. Regardless of diversity, full cooperation requires more computation time per function evaluation than partial and no cooperation, which are on par with one another in this respect.

Again, the results in Fig. 2 are for structurally identical cases of PMAS—when the number of iterations to run and the population size are specified. To compare the solution quality of different classifications for *equal* computational times, Pareto frontiers were generated for the extreme classifications using solution quality and computation time as objectives. Figure 3 plots the range of expected objective function values (the mean and standard error of 50 trials of PMAS) for the heterogeneous, fully dynamic, and fully cooperative case (blue/dark gray) and for the homogeneous, static, and noncooperative case (pink/light gray) for given computational times. Prior to about 1 s in these cases, it does not matter whether a practitioner chooses a homogeneous, static, and noncooperative approach or a heterogeneous, fully dynamic, and fully cooperative approach. After the 1 s mark, however, there is a clear difference between the two approaches in terms of solution quality. The heterogeneous, fully dynamic, and fully cooperative approach quickly converges to the optimum solution, whereas the homogeneous, static, and noncooperative approach hovers around a suboptimal solution.

6.2 Three-Dimensional Packing. The results for running structurally identical trials of PMAS with different classifications on the 3D packing problem, shown in Fig. 4, are similar to those of the one-max problem. For example, homogeneous approaches, which have low computational costs, also have low solution quality. The variances for homogeneity are also larger than those for heterogeneity. As in one-max, no cooperation is less computationally expensive than full cooperation, but with a trade-off of poorer solution quality. Though differences between levels of dynamism are not as clear for 3D packing, in most cases either the structure or fully dynamic implementation has the best solution quality of the four levels or the two are equivalent to the best level.

These initial results are, again, for structurally identical runs of PMAS. For a given computational time, the expected objective function value for a homogeneous, static, and noncooperative implementation versus a heterogeneous, fully dynamic, and fully cooperative implementation is given in Fig. 5. As in the one-max problem, the initial solution quality for low computation times is about the same for both implementations. After this initial similarity, however, the heterogeneous, fully dynamic, and fully cooperative case converges to better solution quality than that found in the homogeneous, static, and noncooperative implementation.

The most striking aspect of the results of structurally identical 3D packing runs provided in Fig. 4 is their qualitative similarity to those of the structurally identical one-max runs given in Fig. 2. For example, at the low objective function value, high computa-

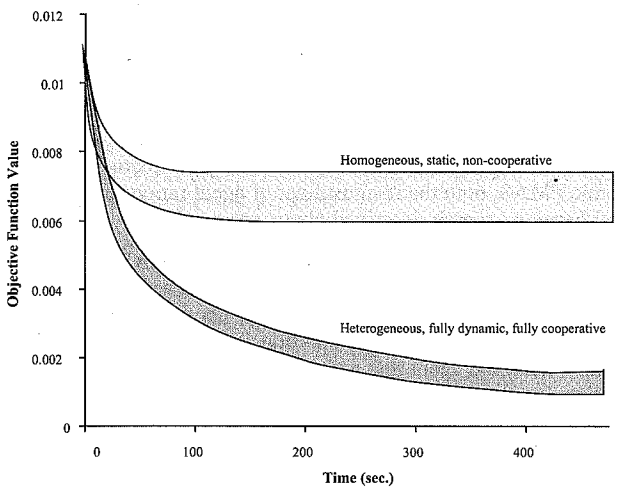

Fig. 5 Computation time versus mean and standard error of objective function value when both a heterogeneous, fully dynamic, and fully cooperative approach and a homogeneous, static, and noncooperative approach are implemented in PMAS and applied to the three-dimensional packing problem

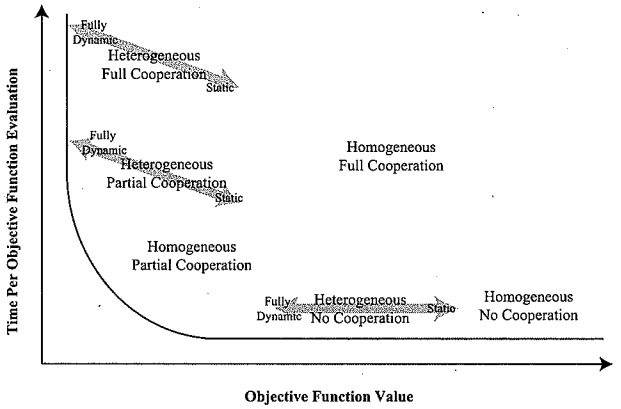

Fig. 6 Emergent trends and frontier based on PMAS data

tion time region of the graph, there are primarily heterogeneous partial and full cooperation implementations. The ones with the best solution quality in these groups are the structure and fully dynamic implementations, which correspond to the upper right of the taxonomy. At the low computation time, high objective function value region are noncooperative approaches. Between these extremes are partially cooperative homogeneous approaches. Finally, dominated by all other groups are the homogeneous fully cooperative scenarios. An illustration of these trends is given in Fig. 6. All of the data oriented with respect to the "frontier" are shown in Fig. 7. This frontier may be likened to a Pareto frontier of structurally identical PMAS cases; however, in generating the data, neither the time per objective function evaluation nor the objective function value was intentionally varied during the experiments. While there are a few outliers (circled in Fig. 7), the data generally fit the trends given in Fig. 6. Note that these trends are qualitative and based on observation of the data, rather than quantitative.

The trends support those of previous EMAS work and the hypothesis posed by this work—that cooperative and highly dynamic approaches have better solution quality than noncooperative and less dynamic approaches. On average, the solutions produced by implementations in the upper right corner of the taxonomy are 55% better in terms of solution quality than non-upper-right implementations for the one-max problem and 63% better

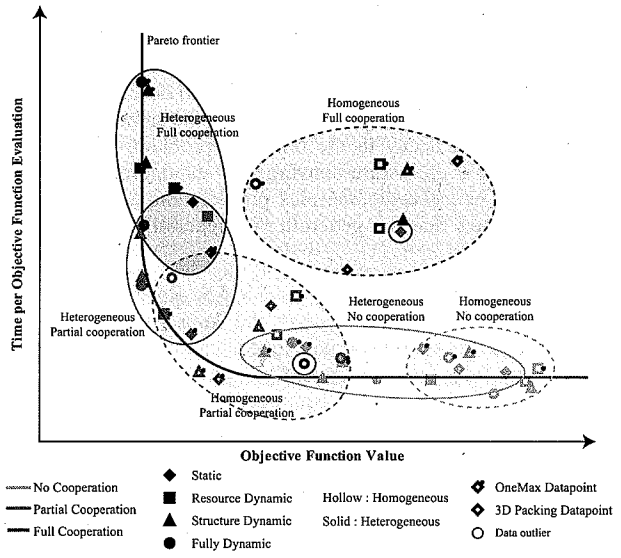

Fig. 7 One-max and 3D layout data with frontier overlaid and classifications from Fig. 1 indicated

for the 3D packing problem in the structurally identical case.[1] For example, the classification corresponding to the best available extensions to simulated annealing and pattern search (heterogeneous, structure dynamic, and noncooperative) is more than 90% worse than the partially cooperative, heterogeneous, and structure dynamic implementation for both problems when the number of agents and the number of iterations are prespecified. Furthermore, approaches which use many different strategies have better solution quality than those that use only a single strategy. Within the cooperative and highly dynamic implementations, heterogeneous approaches are, on average, around 89% better than homogeneous approaches for the 3D packing problem and 99% better on average for the one-max problem.

The trade-off to this high solution quality, in many cases, is computation time. High dynamism and cooperation require, on average, 42% higher computation time than the rest of the taxonomy for the one-max problem and 23% higher computation time for the 3D packing problem for the structurally identical case. Even so, Figs. 3 and 5 illustrating the individual Pareto frontiers of the two approaches at the extremes of the taxonomy indicate that heterogeneous, fully dynamic, and fully cooperative approaches quickly overtake those that are homogeneous, static, and noncooperative in terms of solution quality as allowed computation time increases.

6.3 Importance of Generation Versus Selection. Figures 2 and 4 illustrate the data resulting from identical classifications for both generating and selecting protocols. Recall from Sec. 2.4 that generating and selecting protocols may be classified differently, so an additional experiment was performed to examine their individual influences in structurally identical PMAS runs. In these experiments, either the generating protocol or the selecting protocol classification was held at the best implementation (as identified by the experimentation of Secs. 6.1 and 6.2), and the other was varied across all implementations.

For the one-max problem, the difference between the best and worst classifications is more than 18 times greater when the generating protocol classification is changed than when the selecting protocol classification is changed. The difference between the longest and shortest times per function evaluation is 90% greater when the generating protocol classification is changed. This would suggest that the generating protocol classification is more influential on both solution quality and computation time than the selecting protocol classification, at least when the opposite protocol's classification is held at its best level.

For the 3D packing problem, the difference between the best and worst classifications is found to be 50% greater when the generating protocol classification is changed than when the selecting protocol classification is changed. Though not as significant as the differences identified between protocols in the one-max problem, this also suggests that the generating protocol classification is more influential on solution quality than the selecting protocol classification. Also like one-max, the difference between the longest and shortest times per function evaluation is 10% greater when the generating protocol classification is changed than when the selecting protocol classification is changed. While the influence of the generating protocol classification appears to outweigh that of the selecting protocol classification in these two experiments, further examination is necessary to determine more precisely the degree of interaction between the different protocol classifications, particularly with respect to different problem domains.

6.4 Generating and Selecting Structure Trends. In the structure dynamic and fully dynamic implementations of PMAS, the structure of the agents, in addition to their parameters, changes over time. In earlier work on EMAS, it was shown that some

[1]Calculated by taking the average solution quality of all runs of PMAS implementations that are in the upper right corner of the taxonomy versus the average solution quality of all runs of other PMAS implementations.

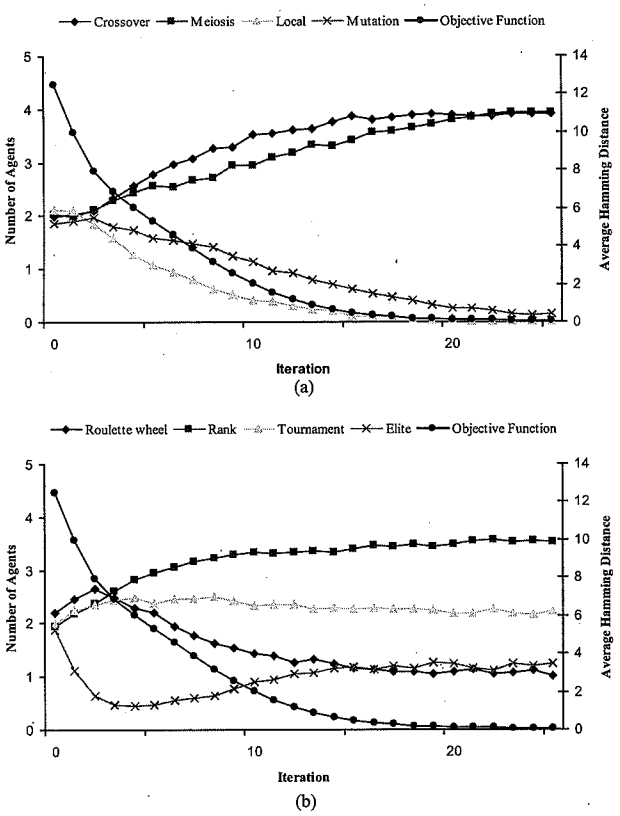

Fig. 8 (*a*) Generating agent and (*b*) selecting agent population trends in fully dynamic PMAS executed on the one-max problem

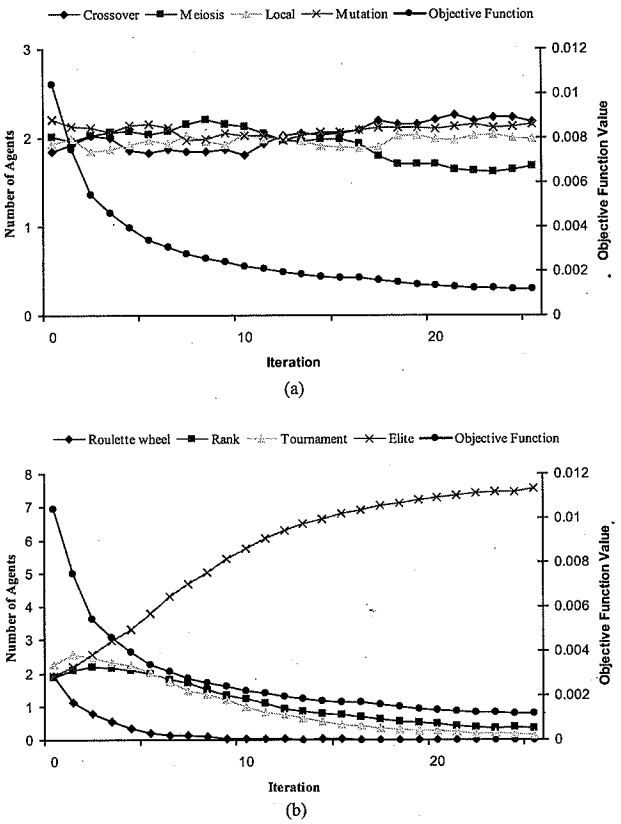

Fig. 9 (*a*) Generating agent and (*b*) selecting agent population trends in fully dynamic PMAS executed on the 3D packing problem

algorithms are consistently more successful at different times in the optimization. To determine whether similar patterns emerged in PMAS, 100 trials of both the one-max problem and the 3D packing problem were run for the heterogeneous, fully cooperative, and fully dynamic scenario. The numbers of each type of agent in the one-max implementation are shown in Fig. 8 (left axis) with the objective function trend overlaid on the team data (right axis) and indicate that patterns do indeed emerge. The results shown in Fig. 8(*a*) indicate that generating agents running crossover and meiosis are more successful than those running mutation and local search since the number of agents running these protocols generally increase over time. Crossover is slightly more successful than meiosis, and mutation is slightly more successful than local search, but these differences are minor. The results shown in Fig. 8(*b*) indicate that for selection, roulette wheel selection is perhaps helpful initially since more roulette wheel agents appear early on, but rank-based selection is more effective later on (around the sixth iteration in this case). Elitist selection is found to be one of the worst protocols for one-max, which could be explained by the incorporation of the uniqueness measure into the fitness function of the selecting agents (see Eq. (2)). Since there are only so many binary strings, especially at very low hamming distances, elitist agents may always choose the same one, which would decrease their fitness.

As shown in Fig. 9, different agent number trends are observed for the 3D packing problem (again, the objective function is overlaid on the graph, and its value is given by the right axis). No generating protocol distinguishes itself from any of the other generating protocols (Fig. 9(*a*)), indicated by the fact that the number of each agent type averages out to be approximately the same throughout the optimization. The selecting team characteristics, shown in Fig. 9(*b*), are also very different from those of the one-max problem. Elitist selection dominates in the 3D packing problem, whereas fewer agents run roulette wheel, rank, and tournament selection as the algorithm progresses. Tournament selection is helpful initially but quickly decreases in prevalence, as does rank-based selection, and roulette wheel selection falls off quickly after the start of the optimization. This is surprising given the lack of elitist agents in the one-max problem. These results suggest that for highly complex problems such as 3D packing, direct search methods with greedy selecting protocols, such as pattern search, may be more effective than searches with more exploratory selecting protocols. This is counterintuitive given the highly multimodal nature of the space, but the fact that none of the generators distinguish themselves suggests that they are all equally important, so direct selection should be coupled with very different modes of generation to ensure that the space is explored to the fullest extent. Notice that the objective function continues to decrease in the 3D layout problem, even though initially it has the same shape as the objective function decay in the one-max problem, which is encouraging from a premature-convergence standpoint. The difference in trends between the one-max problem and the 3D packing problem indicates that the generality of generating and selecting trends is limited.

6.5 Comparison of PMAS to EMAS. Recall that another goal of this work was to remedy the long computation time needed to run the previously developed heterogeneous, structure dynamic, and fully cooperative EMAS algorithm. In this section, the computation time for the heterogeneous, structure dynamic, and fully cooperative implementation of PMAS is compared with the computation time of EMAS. The average computation time for this classification of PMAS for 25 iterations on the 3D packing problem is approximately 8 minutes. Of the 50 PMAS trial runs, 30% are found to reach the same or better solution quality as the average found when 25 iterations of EMAS are run on the 3D packing problem, though a single trial of EMAS takes several hours. Furthermore, even the worst trials of PMAS reach the so-

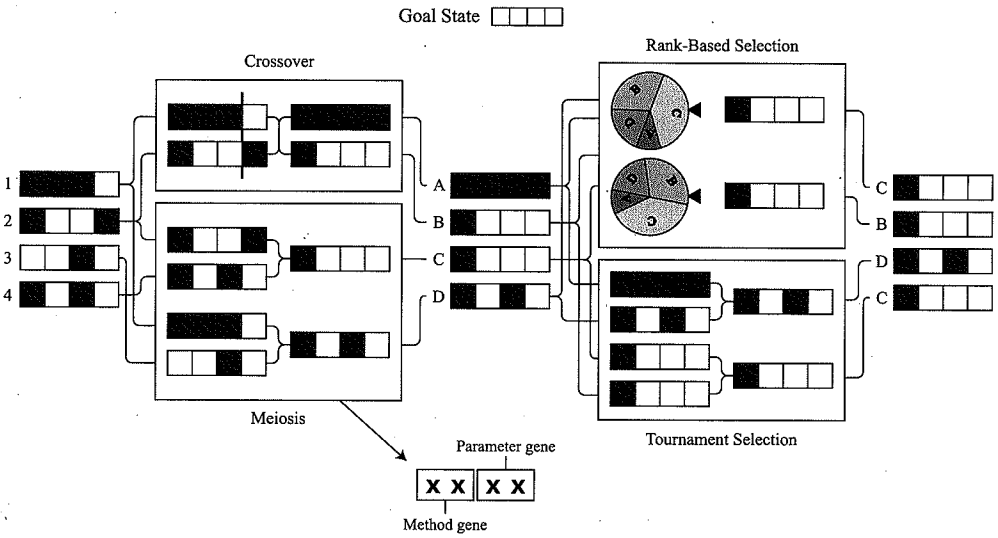

Fig. 10 Example of single iteration of heterogeneous, fully dynamic, and fully cooperative PMAS

lution quality found by EMAS in under 30 min. This constitutes a more than 95% speed-up over the EMAS algorithm without sacrificing solution quality.

7 Conclusions

A taxonomy of heuristic optimization approaches was presented, and a review of the literature indicated that relatively few approaches simultaneously employ many different search strategies, allow those strategies to adapt to the problem, and promote cooperation between strategies. Based on the literature and some results of recent work in this area, it was hypothesized that approaches with these characteristics are likely to produce high quality solutions to complex engineering problems.

The evolutionary multi-agent systems framework was expanded to form the protocol-based multi-agent systems architecture, in which diversity, dynamism, and cooperation can be examined at the protocol level rather than at the algorithm level. Adapting and combining protocols in PMAS was found to lead to approximately the same solution quality as combining and adapting algorithms in EMAS but in a fraction of the time. Each classification in the taxonomy was explored at this finer resolution, and it was found that even for very different problems, the trends for solution quality and computation time are similar when the structure of PMAS (the number of iterations and the agent population size) is held constant. For instance, cooperation led to better performance for both the one-max problem and for a complex 3D packing problem than no cooperation, but at the cost of more computation time. Similarly, heterogeneity is found to be superior to homogeneity in terms of solution quality but does not significantly increase computation time. Finally, the structure dynamic and fully dynamic implementations have the best performance in terms of solution quality but in some cases require more computation time than static and resource dynamic implementations. These dynamism trends are less pronounced for the 3D packing problem than for the one-max problem. Similar trends in solution quality and computation time also exist when the selecting protocol classification is held constant, indicating that the generating protocol classification is more influential on both computation time and solution quality than the selecting protocol classification, especially for complex problems. When comparing the Pareto frontiers of the two extreme points of the taxonomy, it is found that more computation time for the homogeneous, static, and noncooperative approach does not lead to the better solution quality of the heterogeneous, fully dynamic, and fully cooperative approach.

From the analysis of which agents are most successful at any point in the optimization process, the results for the one-max problem are found to be very different from those of the 3D packing problem. No generating agents significantly distinguished themselves from any other in the 3D packing problem, whereas in the one-max problem it is clear that crossover and meiosis are most effective. Also, while elitist selection is clearly dominant in the 3D packing problem, it is one of the worst selecting protocols in the one-max problem, where rank-based selection is most effective. Some of the one-max trends may be explained by the fact that uniqueness is used to evaluate selecting agents, but for 3D layout, these trends suggest that direct selection, paired with highly diverse generation, helps overcome highly multimodal problem spaces. This supports the hypothesis that the best search strategy for a particular engineering problem changes unpredictably over the course of the optimization and illustrates that different problems require different strategies.

In addition to finding evidence for the hypothesis proposed earlier, this work has also introduced a novel optimization approach capable of producing high quality solutions to complex optimization problems. The heterogeneous, fully dynamic, and fully cooperative implementation of PMAS (hff-PMAS) consistently generated solutions of high quality, especially compared with other PMAS implementations—including those analogous to algorithm portfolios and traditional simulated annealing algorithms. Thus, hff-PMAS, which embodies multiple protocols in independent agents and allows those agents to share all solutions with one another and adapt their structure and resources to the problem at hand, may be used by itself as a viable optimization approach. Because hff-PMAS increases the efficacy of each strategy used within it and further adjusts strategies to the problem as it is being solved, practitioners can spend less time fine-tuning a single strategy used throughout the optimization.

Appendix: Example of PMAS Implementation

To illustrate how diversity, dynamism, and cooperation are implemented in PMAS, the following example of the heterogeneous, fully dynamic, and fully cooperative PMAS implementation is provided. Figure 10 illustrates a single iteration of the PMAS algorithm in the context of the one-max problem with $n=4$. In this example, at the start of the iteration, four solutions, 1–4, are fed to two generating agents. One generating agent embodies crossover, and the other embodies meiosis. Because two different generating strategies are available at the same time, the

approach is heterogeneous. In a homogeneous case, both agents would be identical to one another (though the one identical strategy used by both could potentially change over time depending on the level of dynamism).

To implement full cooperation, both agents choose solutions from the same pool, here, solutions 1–4. In a noncooperative scenario, each generating agent would have a different pool of solutions available to use (solutions in its individual inbox). Similarly, in a partially cooperative scenario, agents would have mostly distinct solutions to choose from but one or more solutions available to use that are also available to other agents. The nature of crossover is to create two solutions from two input solutions, and in this example the crossover agent uses solutions 1 and 2 to create solutions A and B. The nature of meiosis is to create one solution from two input solutions, so to create two solutions (the same number as the crossover agent), C and D, the meiosis agent acts twice using solutions 2 and 4, and then 1 and 3. The solutions created by the generating agents, A–D, are subsequently subjected to selection by the selecting agents.

In this example, there are two selecting agents, embodying rank-based selection and tournament selection. Again, because the two selecting strategies differ, the approach is heterogeneous. Also, because they select from the same pool of solutions, the approach is fully cooperative. The rank-based selecting agent selects solutions C and B from the four solutions, and the tournament selecting agent chooses solutions D and C. These selected solutions would then be passed to the next iteration's generating agents, and the process would continue.

Now, in a static or resource dynamic scenario, the next iteration of generating and selecting agents would be identical to those in the previous iteration (crossover and meiosis, rank-based and tournament selection). In structure or fully dynamic scenarios, the agents would be represented by chromosomes, which would be subjected to the harmony search operator described in Sec. 5.2. The fitness of the agents, evaluated in nonstatic scenarios, is calculated using Eqs. (1) and (2). In this example, the crossover agent would have a fitness of $1/2=0.5$ since only one of its two solutions was selected, and the meiosis agent would have a fitness of $2/2=1$. The rank-based selecting agent would have a fitness of $1/2=0.5$ since it selected only one unique solution of two solutions selected, and the tournament selection agent would also have a fitness of $1/2=0.5$ after selecting only one solution better than or equal to the median objective function value of solutions C, B, C, and D (median objective function value=3).

In redistributing resources for resource dynamism, Eq. (3), with $r_{\min}=1$, specifies that the crossover agent would get one solution to generate the next iteration and meiosis would get three. Since both selecting agents have the same rank of 0, Eq. (3) specifies that each would be allowed to generate just one solution the next iteration. However, since the total number of solutions selected must equal 4, two solutions will be randomly allotted to the selecting agents. This means that either both will receive two solutions to select or one will have only one solution and the other will have three. In creating new structures, a new generating agent would be created in the next iteration to replace the crossover agent according to the harmony search protocol mentioned in Sec. 5.2. Likewise, a new selecting agent would be created and randomly replace either the rank-based or tournament selecting agent. Note that in a homogeneous scenario, the new agent created would be replicated so that all agents had identical structures.

References

[1] Haupt, R. L., and Haupt, S. E., 2004, *Practical Genetic Algorithms*, Wiley, Hoboken, NJ.

[2] Geem, Z. W., 2009, "Music-Inspired Harmony Search Algorithm: Theory and Applications," *Studies in Computational Intelligence Series*, Springer-Verlag, Berlin.

[3] Thierens, D., 2002, "Adaptive Mutation Rate Control Schemes in Genetic Algorithms," Proceedings of the 2002 IEEE World Congress on Computational Intelligence: Congress on Evolutionary Computation.

[4] Cantu-Paz, E., 1997, "A Survey of Parallel Genetic Algorithms," University of Illinois at Urbana-Champaign, Technical Report No. 97003.

[5] Gomes, C. P., and Selman, B., 2001, "Algorithm Portfolios," Artif. Intell., **126**, pp. 43–62.

[6] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," Science, **220**, pp. 671–680.

[7] Torczon, V., and Trosset, M. W., 1998, "From Evolutionary Operation to Parallel Direct Search: Pattern Search Algorithms for Numerical Optimization," Computing Science and Statistics, **29**, pp. 396–401.

[8] Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., and Shoham, Y., 2003, "Boosting as a Metaphor for Algorithm Design," Lect. Notes Comput. Sci., **2833**, pp. 899–903.

[9] Petrik, M., and Zilberstein, S., 2006, "Learning Parallel Portfolios of Algorithms," Ann. Math. Artif. Intell., **48**, pp. 85–106.

[10] Streeter, M., Golovin, D., and Smith, S. F., 2007, "Combining Multiple Heuristics Online," Proceedings of the 22nd Conference on Artificial Intelligence, Vol. 2, pp. 1197–1203.

[11] 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.

[12] 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, pp. 286–293.

[13] 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**, pp. 172–192.

[14] Back, T. D., Hoffmeister, F., and Schwefel, H. P., 1991, "A Survey of Evolution Strategies," Proceedings of the Fourth International Conference on Genetic Algorithms.

[15] Pohlheim, H., 2001, "Competition and Cooperation in Extended Evolutionary Algorithms," Proceedings of the Genetic and Evolutionary Computation Conference.

[16] Huberman, B. A., Lukose, R. M., and Hogg, T., 1997, "An Economics Approach to Hard Computational Problems," Science, **275**, pp. 51–54.

[17] Fogarty, T., 1989, "Varying the Probability of Mutation in the Genetic Algorithm," Proceedings of the Third International Conference on Genetic Algorithms, pp. 104–109.

[18] Hesser, J., and Manner, R., 1991, "Towards an Optimal Mutation Probability in Genetic Algorithms," Proceedings of the First Parallel Problem Solving From Nature.

[19] Hesser, J., and Manner, R., 1992, "Investigation of the M-Heuristic for Optimal Mutation Probabilities," Proceedings of the Second Parallel Problem Solving From Nature.

[20] Back, T., and Schutz, M., 1996, "Intelligent Mutation Rate Control in Canonical Genetic Algorithms," Proceedings of the International Symposium on Methodologies for Intelligent Systems.

[21] Yin, S., and Cagan, J., 2000, "An Extended Pattern Search Algorithm for Three-Dimensional Component Layout," ASME J. Mech. Des., **122**, pp. 102–108.

[22] 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.

[23] Gagliolo, M., and Schmidhuber, J., 2004, "Adaptive Online Time Allocation to Search Algorithms," Lect. Notes Comput. Sci., **3201**, pp. 134–143.

[24] Gagliolo, M., and Schmidhuber, J., 2007, "Learning Dynamic Algorithm Portfolios," Technical Report No. IDSIA-02-07.

[25] Moral, R. J., Sahoo, D., and Dulikravich, G. S., 2006, "Multi-Objective Hybrid Evolutionary Optimization With Automatic Switching," 11th AIAA Multidisciplinary Analysis and Optimization Conference.

[26] Moral, R. J., and Dulikravich, G. S., 2008, "Multi-Objective Hybrid Evolutionary Optimization With Automatic Switching Among Constituent Algorithms," AIAA J., **46**, pp. 673–681.

[27] Dorigo, M., and Stutzle, T., 2004, *Ant Colony Optimization*, MIT, Cambridge, MA.

[28] 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.

[29] Smith, J., and Fogarty, T. C., 1997, "Operator and Parameter Adaptation in Genetic Algorithms," Soft Comput., **1**, pp. 81–87.

[30] Szykman, S., and Cagan, J., 1995, "A Simulated Annealing-Based Approach to Three-Dimensional Component Packing," ASME J. Mech. Des., **117**, pp. 308–314.

[31] Hoffmeister, F., and Back, T., 1991, "Genetic Algorithms and Evolution Strategies: Similarities and Differences," Lect. Notes Comput. Sci., **496**, pp. 455–469.

[32] Cantú-Paz, E., 2001, "Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms," J. Heuristics, **7**, pp. 311–334.

[33] Talukdar, S., Baerentzen, L., Gove, A., and DeSouza, P., 1998, "Asynchronous Teams: Cooperation Schemes for Autonomous Agents," J. Heuristics, **4**, pp. 295–321.

[34] Vrugt, J. A., and Robinson, B. A., 2007, "Improved Evolutionary Optimization From Genetically Adaptive Multi-Method Search," Proc. Natl. Acad. Sci. U.S.A., **104**, pp. 708–711.

[35] Nii, H. Y., 1986, "Blackboard Systems," Stanford University, Technical Report No. STAN-CS-86-1123/KSL-86-18.

[36] 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 Conferences and Computers and Information in Engineering Conference IDETC/CIE.
[37] Hanna, L., and Cagan, J., 2009, "Evolutionary Multi-Agent Systems: An Adaptive and Dynamic Approach to Optimization," ASME J. Mech. Des., **131**, p. 011010.
[38] Aladahalli, C., Cagan, J., and Shimada, K., 2007, "Objective Function Effect Based Pattern Search—Theoretical Framework Inspired by 3D Component Layout," ASME J. Mech. Des., **129**, pp. 243–254.
[39] Hohn, C., and Reeves, C., 1996, "The Crossover Landscape for the Onemax Problem," Proceedings of the Second Nordic Workshop on Genetic Algorithms.
[40] Mortensen, M. E., 1997, *Geometric Modeling*, Wiley, New York.