1 INTRODUCTION

Focused research has uncovered mechanisms of design cognition and revealed insights that can be used to inform the methods used by human designers (Finger and Dixon, 1989; Cross, 2004; Cagan et al., 2013). Using these principles of design cognition to inform computational methodologies has the potential to improve the performance of design algorithms. This paper draws on work that modeled the performance of individual problem-solvers via the simulated annealing (SA) algorithm (Cagan and Kotovsky, 1997). However, rather than use simulated annealing as a model of human problem-solving, the current work explores the possibility of using insights from design research to offer improvements to existing optimization algorithms. Specifically, this work draws upon the behavior and composition of engineering design teams as a useful analog for suggesting improvements to the class of simulated annealing optimization algorithms.

A number of studies have sought to draw connections between human designers and computational optimization and design tools. For instance, neural networks have been used in conjunction with genetic algorithms in an attempt to allow computers to consider stylistic aspects of design, much like human designers would (Tseng et al., 2012). Some agent-based design algorithms even attempt to approximate the differing design approaches and knowledge that are beneficial in human design teams (Campbell et al., 1999). In another example, insights from empirical studies of analogy use were used to design a computational system that was intended to assist human designers (Goel et al., 2012). Work by Egan et al. (2014) demonstrated the potential benefit of using computational agents to rapidly test and refine search strategies that can then be provided to human designers, closing the synergistic loop between computer and designer.

This work centers around SA, a heuristic optimization algorithm that begins with stochastic exploration, and progressively transitions towards deterministic search (Kirkpatrick et al., 1983). Individual problem-solvers display a very similar transition, allowing SA to effectively model the human solving process (Cagan and Kotovsky, 1997). Therefore, SA methodology is a rational starting point for the development of a team-inspired optimizing search algorithm. The SA algorithm is inspired by the physical annealing process, in which materials are heated and cooled in a controlled manner to minimize residual stresses. When the material is hot, the movement of atoms is random in nature; atoms may even move occasionally in a direction that *increases* potential energy. As the material is cooled, atomic movements become more deterministic, and atoms begin to move almost entirely in directions that decrease potential energy. In the analogous optimization algorithm, the goal is to minimize the value of the objective function, rather than potential energy. An initial solution is chosen at random, and a new solution is proposed in every subsequent iteration. If the new solution has a better objective function value than the previous solution, it is accepted. If not, the new solution may still be accepted with some probability. The probability of accepting a worse solution is typically decreased after every iteration, enabling the algorithm to progressively transition from initial stochastic search to final deterministic search.

An important consideration in any SA algorithm is the annealing schedule. The annealing schedule dictates the simulated temperature, which in turn dictates the probability of accepting a worse solution. The temperature may also influence the generation of new solution candidates in some SA algorithms. The most rudimentary annealing schedules are monotonically decreasing functions of iteration number. Theoretically, the probability that the algorithm will find the global optimum approaches unity as the annealing schedule is stretched over an increasing number of iterations (Granville et al., 1994). However, computational resources are finite, which makes the use of extremely long annealing schedules impractical. Therefore, the implementation of adaptive annealing schedules has been of particular interest to simulated annealing practitioners. For instance, a schedule proposed by Azizi and Zolfaghari (2004) follows a conventional geometric annealing schedule, but increases the temperature for every consecutive uphill move that is made. The intuition is that this will help the algorithm to escape local minima. This adaptive schedule was found to perform better than conventional annealing schedules on a job-scheduling task. Triki et al. (2005) demonstrated that many common adaptive temperature schedules follow very similar forms, and identified a single term within the temperature update rule that differentiated between schedules. While other schedules defined this term using a function of objective function variance, or number of accepted solutions, Triki et al. (2005) proposed a new schedule that simply replaced the term with a constant. This schedule performed better than

another adaptive schedule by Huang et al. (1986) on a variety of benchmarking tasks. Adaptive schedules can be made even more responsive to the problem space by implementing re-annealing (boosting the temperature rapidly when a local minimum has been found) and quenching (quickly decreasing the temperature when a local minimum is likely).
A number of algorithms have been proposed that run several SA algorithms as parallel sub-routines, and combine solutions at regular intervals using computational genetic operators (Hiroyasu et al., 2002; Ohlidal and Schwarz, 2004). More recent work has developed Multi-agent Simulated Annealing (MSA) algorithms, which employ software agents to operate on multiple solutions. Within this context, a software agent, usually referred to simply as an agent, is a computational sub-routine that operates on potential solutions with some degree of autonomy. The MSA algorithms utilize principles of differential evolution (Zhong et al., 2012a) and particle swarm optimization (Zhong et al., 2012b) to accomplish interaction between agents. However, the agents used in these algorithms do not possess any sort of individual strategy or preference for exploring solutions. The ownership of personal characteristics is a common factor in many agent-based algorithms (Franklin and Graesser, 1997). It can also lead to heterogeneity and diversity of agents, which has been demonstrated to improve performance in agent-based optimization models (Landry and Cagan, 2011). Creating diversity between agents in an SA-based algorithm may improve performance.
Although a team is composed of individual problem-solvers, there is often additional benefit that is derived from interaction between the individuals (Wood et al., 2012). This arises from the ability of a team to initially diverge to explore a variety of options, but then converge, focusing the attention of the team members on a shrinking set of alternatives (Fu et al., 2010). Even members of high-performing teams tend to pursue slightly different solution concepts while solving well-defined problems (McComb et al., 2014), indicating that members of a team don't always greedily pursue the solutions with highest quality. Therefore, though team members may factor design quality into decisions, they freely pursue designs that may currently display lower quality. Further, it is known that expert designers tend to use a mixture of depth- and breadth-first solution strategies (Ball and Ormerod, 1995). Therefore, in a design team it becomes important for designers to have the ability to asynchronously employ the search strategies that they individually deem best, leading to the emergence of heterogeneous strategies within a team.
This work introduces the Heterogeneous Simulated Annealing Team (HSAT) algorithm. This algorithm provides each agent with an independently controlled, adaptive annealing schedule, allowing agents to develop heterogeneous search strategies. The potential benefit of heterogeneity has been demonstrated in other heuristic optimization algorithms, and can also be justified through analogical comparison to human design teams. Further, interaction between agents in HSAT is designed to mimic characteristics of interaction in human design teams. The HSAT algorithm is compared to a non-adaptive SA algorithm, an adaptive SA algorithm, a non-adaptive multi-agent SA algorithm, random search, and a gradient-based algorithm. Performance of these algorithms is compared on two benchmarking functions.

2 SIMULATED ANNEALING ALGORITHMS

The HSAT algorithm builds upon the conventional SA algorithm, which will be introduced first. Then, the HSAT algorithm will be explained in detail.

2.1 Conventional Simulated Annealing Algorithm

A conceptual flowchart of the conventional SA algorithm is provided in Figure 1. With the initialization of the algorithm, an initial solution candidate is generated (either at random or using a some sort of heuristic). The algorithm then iterates to improve the solution. Within each iteration, a new solution candidate is generated. It is then accepted or rejected according to its objective function value and the temperature of the annealing schedule. The temperature is then updated, and the algorithm continues to the next iteration if convergence criteria have not been met.

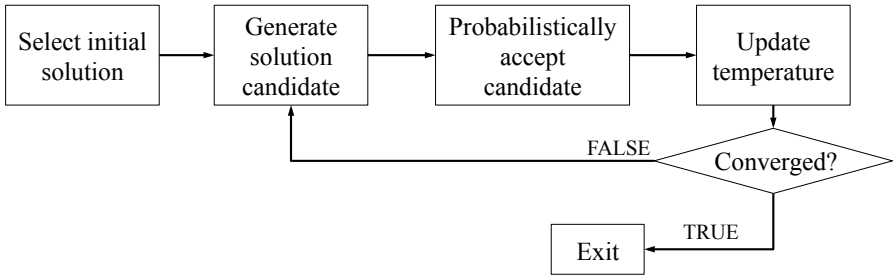


Figure 1. Generalized flowchart for conventional SA.

In this work, initial solutions are selected uniformly at random within a continuous space that is defined by some upper and lower bounds. The values of the bounds are specific to the problem being solved. A new candidate solution, $\vec{\boldsymbol{x}}_{new}$, is created by drawing a solution at random from the Cauchy distribution and adding the resulting vector to the current solution, $\vec{\boldsymbol{x}}_i$. This is accomplished by computing

$$\vec{\boldsymbol{x}}_{new} = \vec{\boldsymbol{x}}_i + T_i \tan\big(\text{uniform}(-\pi/2,\ \pi/2,\ D)\big), \tag{1}$$

where T_i is the current temperature. The function uniform draws a point at random from the continuous D-dimensional space with an upper bound of $\pi/2$ and a lower bound of $-\pi/2$ in each direction. The Cauchy distribution is preferable to the Gaussian distribution because it has thicker tails, and thus encourages more extensive search (Ingber, 1996). Once a new solution is generated, the objective function is evaluated. If the solution is better than the previous solution, is it accepted ($\vec{\boldsymbol{x}}_{i+1} \leftarrow \vec{\boldsymbol{x}}_{new}$). If the new solution is *not* better than the previous solution, is it still accepted with some probability *p*, which is defined as

$$p = \exp\left(\frac{f(\vec{\boldsymbol{x}}_{new}) - f(\vec{\boldsymbol{x}}_i)}{T_i}\right), \tag{2}$$

where $f(\vec{\boldsymbol{x}})$ is the objective function. If the new solution is not accepted, the previous solution is carried into the next iteration ($\vec{\boldsymbol{x}}_{i+1} \leftarrow \vec{\boldsymbol{x}}_i$). Two annealing schedules are used in this work: the classical Cauchy schedule, and the adaptive schedule proposed by Triki et al. (2005). For the Cauchy schedule the temperature is updated as

$$T_{i+1} = \frac{T_0}{1+\delta_C \cdot i}, \tag{3}$$

where T_0 is the initial temperature, i is the index of the current iteration, and δ_c is a parameter that allows the schedule to be extended or compressed. The second schedule (referred to as the Triki schedule for the remainder of this paper) updates temperature as

$$T_{i+1} = T_i \left(1 - \frac{T_i \cdot \delta_T}{\sigma^2_{f(x)}}\right), \tag{4}$$

where δ_T is a parameter that controls how quickly adaptation occurs, and $\sigma^2_{f(x)}$ is the variance of objective function value of the n most recent accepted solutions. The variable n will be referred to as the memory length.

2.2 Heterogeneous Simulated Annealing Team Algorithm

HSAT is a multi-agent simulated annealing algorithm that draws upon two aspects of human design teams. Every agent in the HSAT algorithm is given an independently adaptive annealing schedule, which allows the agent team to develop heterogeneous and asynchronous search strategies. This is similar to the judicious application of mixed search strategies that is demonstrated by expert human designers (Ball and Ormerod, 1995). Heterogeneity of agents has also been shown to be beneficial in other algorithms (Landry and Cagan, 2011). Interaction between agents in HSAT is implemented in a way that probabilistically encourages agents to pursue better designs, while allowing them to pursue worse designs. This enables agents within a team to pursue slightly different solutions, similar to high-performing human teams studied by McComb et al. (2014). This method of interaction also results in a progressive transition from initial divergence (when agents are making stochastic decisions) to final

convergence (when agents are employing deterministic search). High final convergence (indicative of agreement on a solution) is correlated to final solution quality in human teams (Dong et al., 2004; Fu et al., 2010). A general flowchart is provided in Figure 2 to summarize the HSAT algorithm.

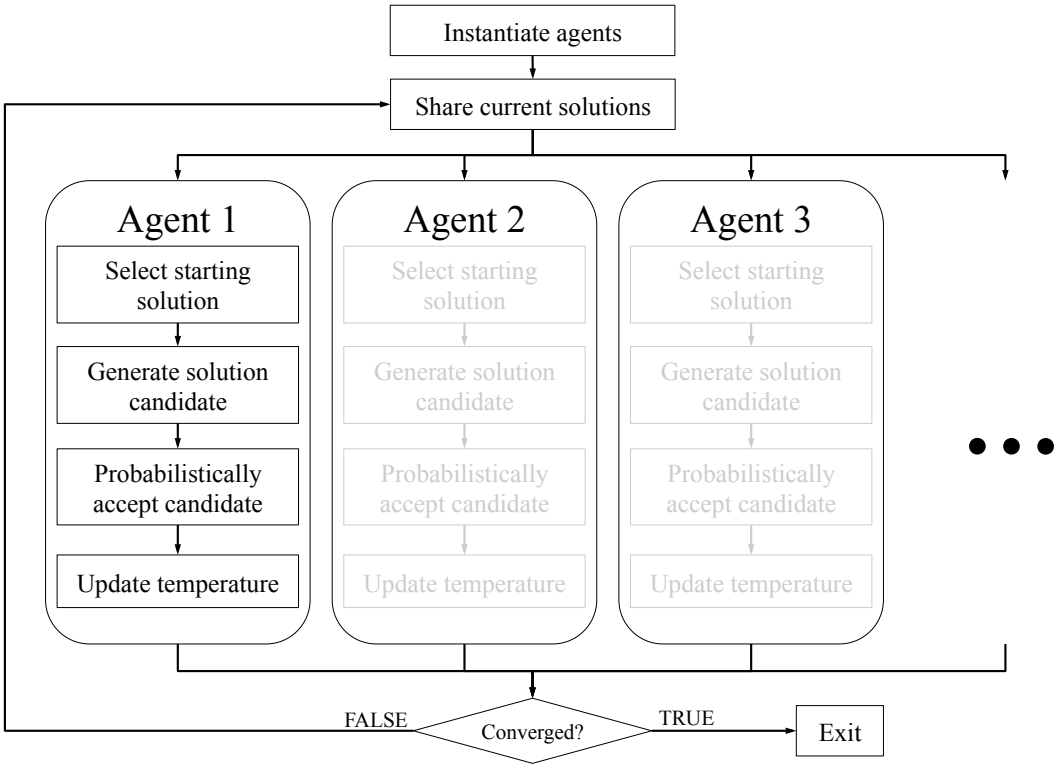


Figure 2. Flowchart for the HSAT algorithm.

When agents are instantiated, each is provided with a candidate solution that is selected uniformly at random. The objective function value of each solution is then shared between agents in the team. Using this information, each agent then selects one of the team's current solutions to begin the iteration from. The objective function value for every agent's current solution is shared through the vector $\vec{\boldsymbol{F}}$,

$$\vec{\boldsymbol{F}} = \left[f\left(\vec{\boldsymbol{x}}_i^1\right), f\left(\vec{\boldsymbol{x}}_i^2\right), \ldots, f\left(\vec{\boldsymbol{x}}_i^N\right)\right]. \tag{5}$$

Note that subscripts indicate iteration number, while superscripts indicate different agents. A new vector $\vec{\boldsymbol{W}}$ is then defined as the relative function value of each current solution compared against the worst current solution:

$$\vec{\boldsymbol{W}} = -\vec{\boldsymbol{F}} + \max(\vec{\boldsymbol{F}}). \tag{6}$$

This equation is only valid for minimization problems, and would need to be modified slightly for maximization. The remaining operations in the iteration are then handled by each agent independently. Each agent selects a starting solution with probability proportional to its relative objective function value, using the equation

$$j = \text{mult}\left(\frac{\vec{\boldsymbol{W}}}{\sum_i W_i}\right), \tag{7}$$

where the function mult returns a draw from a multinomial distribution. This can be thought of as a roll of a loaded die. This procedure provides a means of approximating the interaction of human design teams. Agents in HSAT are probabilistically encouraged to pursue better solutions, but there is still some probability that they will explore worse solutions. The solution selected through this process, $\vec{\boldsymbol{x}}_i^j$, is then used by the agent to begin the next iteration. The equation for calculating the new solution candidate for agent k then becomes:

$$\vec{x}_{new}^{k} = \vec{x}_{i}^{j} + T_i^k \tan\left(\text{uniform}(-\pi/2,\ \pi/2,\ D)\right). \quad (8)$$

In other words, the agent begins at the previous solution $\vec{x}_i^j$, and then applies a Cauchy modification to it, similarly to the conventional SA algorithm. If the new solution candidate, $\vec{x}_{new}^k$, is better than the agent's previous solution, $\vec{x}_i^k$, the solution candidate is accepted. If it is not better, the agent still accepts the solution with acceptance probability computed using Equation 2. Finally, the temperature is updated using the Triki annealing schedule (Equation 4). It should be noted that the temperature is updated independently by each agent, allowing agents to develop heterogeneous strategies.

3 COMPARISON METHODOLOGY

3.1 Compared Algorithms

The algorithm presented in this work differs from the conventional simulated annealing algorithm in two ways: multiple independent agents, and interaction that is informed probabilistically by relative solution quality. Both of these features are observed in high performing human teams. In order to fully understand the impact of these features, HSAT is compared to three other SA-based algorithms as summarized in Table 1.

Table 1. Summary of SA-based algorithms.

	Classical annealing schedule (Cauchy)	Adaptive annealing schedule (Triki)
Single Agent	Non-Adaptive SA: Single-agent simulated annealing algorithm using Cauchy annealing schedule	Adaptive SA: Single-agent simulated annealing algorithm using Triki annealing schedule
Multi-Agent	Non-adaptive MSA: Interaction as in HSAT, but with a classical annealing schedule.	HSAT: Multi-agent simulated annealing using a separate adaptive annealing schedule for *each* agent

Because SA algorithms progressively transition from stochastic to deterministic search, we use a purely stochastic algorithm (random search) and a purely deterministic algorithm (gradient-based) for comparison. The random search algorithm samples randomly within the bounds of the search space at each iteration and returns the best solution encountered. The gradient-based algorithm is the Broyden–Fletcher–Goldfarb–Shanno interior-point algorithm. In the remainder of the paper, this algorithm will be referred to as the *gradient algorithm*. Rather than directly computing the Hessian, this algorithm estimates it using successive gradient information. A more detailed description of the gradient algorithm can be found in Papalambros & Wilde (2000).

Every algorithm (both SA, and non-SA) is permitted 30,000 calls to the objective function. The gradient algorithm is restarted with a new, random location every time it converges on a local minimum, until the allotted number of function evaluations is reached.

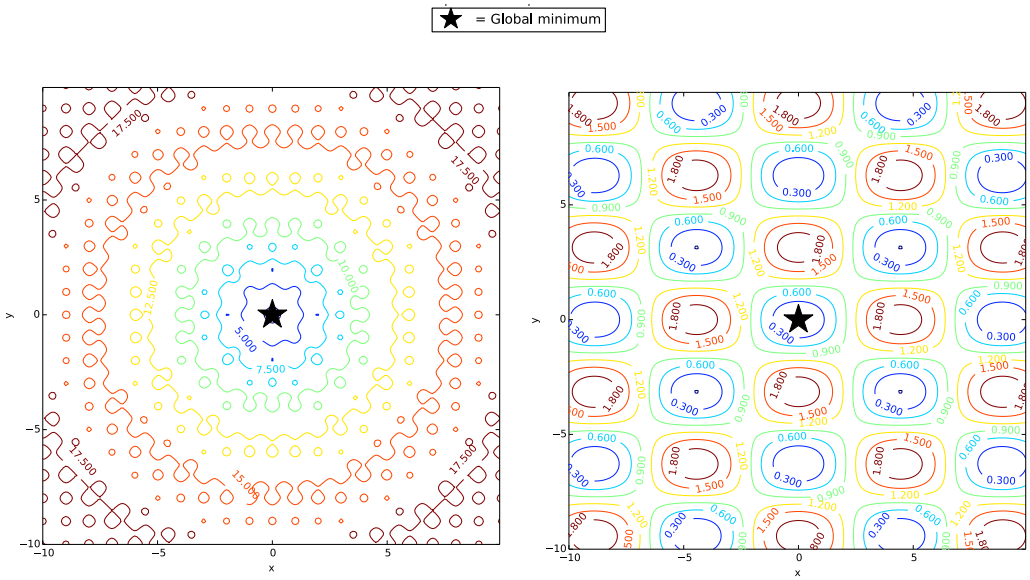
3.2 Benchmarking Functions

Algorithm performance is assessed with respect to two continuous functions. These functions are the Ackley function (Bäck, 1996) and the Griewank function (Griewank, 1981). In this work the fully generalized versions of the functions are used, as shown in Equations 9 and 10. The variable D indicates the number of dimensions in the search space.

$$\text{Ackley:} \qquad f(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i}{D}}\right) - \exp\left(\frac{\sum_{i=1}^{D}\cos(2\pi x_i)}{D}\right) + 20 + \exp(1) \quad (9)$$

$$\text{Griewank:} \qquad f(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (10)$$

Both the Ackley and Griewank functions have their global minimum of $f(\vec{x}^*) = 0$ at $\vec{x}^* = [0, 0, 0, \ldots, 0]$. For the numerical experiments conducted as part of this work, every equation was implemented with 30 dimensions. For both functions, every dimension was constrained so that $-10 \leq x_i \leq 10$. A 2-dimensional representation of each function is provided in Figure 3.



(a) Ackley function. *(b) Griewank function.*

Figure 3. Benchmarking functions.

Both functions present distinct challenges. The Ackley function displays many local minima, but the global minimum lies within a shallow depression that is not accessible from most of the space unless the algorithm can cross numerous small ridges. Further, the global behaviour of the function (disregarding local minima) is non-convex. The Griewank function also displays many local minima, and many are close in objective function value to that of the global minimum (Cho et al., 2008). Relative to the Ackley function, the local minima in the Griewank function are fairly deep, further increasing complexity because the convexity of the quadratic component is obscured. Due to the multi-modal natures of the search spaces of these functions, it is unlikely that an algorithm will efficiently find the global minimum using only gradient information.

3.3 Meta-Optimization of Parameters

The SA-based algorithms compared in this work all differ significantly, either in terms of annealing schedule, number of agents, or both. Consequently, each algorithm has a unique set of parameters that control its execution, including aspects like annealing schedule and the generation of new solutions. Assuming that using the same parameter for every algorithm (for instance, initial temperature) would ensure good performance for all is naïve. In fact, it may be the case that parameter values that are near-optimal for one algorithm may be detrimental to the performance of another. For this reason, a meta-optimization was performance to select the best parameters for each SA-based algorithm. This procedure employed a pattern search to improve the average final solution value of each algorithm by incrementally tuning parameters. Since the average performance must be evaluated over a large number of runs, the basic pattern search algorithm was chosen to decrease computational cost. The parameters defining the annealing schedule were optimized for every algorithm, and for multi-agent algorithms the number of agents was also optimized. The initial step size of the pattern search algorithm was defined as 40% of the initial value of each parameter. If no improvement was found after modifying each parameter in turn, the step size was halved, and the procedure repeated. If no improvement was observed for 5 consecutive iterations, the algorithm terminated (after approximately 20 total iterations).

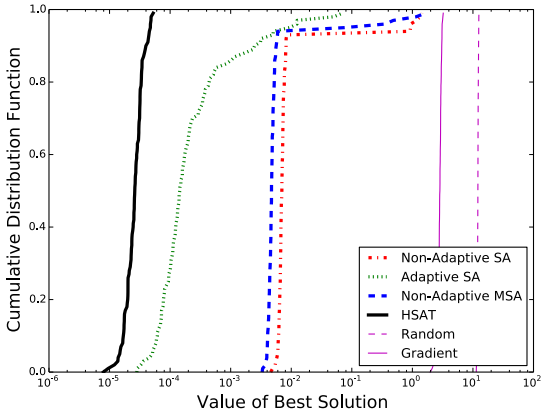
4 RESULTS AND DISCUSSION

Table 2 summarizes the parameters resulting from the pattern search meta-optimization procedure. This procedure was applied to every SA-based algorithm independently on both of the benchmarking functions.

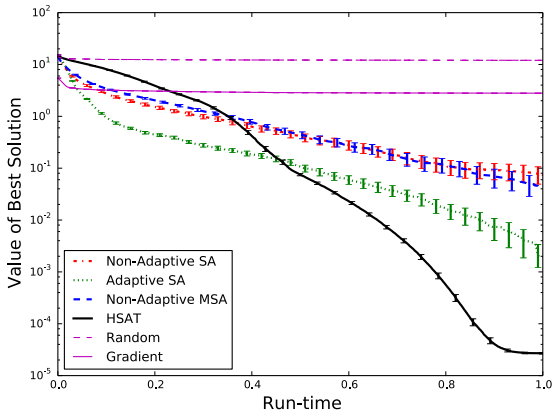
Table 2. Parameters used for SA-based algorithms

Algorithm	Function	Number of Agents, N	Initial Temp., T_0	Cauchy Param., δ_C	Triki Param., δ_T	Memory length, n
Non-Adaptive SA	Ackley	1	0.421	0.037	N/A	N/A
	Griewank	1	0.428	0.026	N/A	N/A
Adaptive SA	Ackley	1	0.047	N/A	5.01×10^{-9}	11
	Griewank	1	0.003	N/A	2.00×10^{-9}	7
Non-Adaptive MSA	Ackley	5	0.900	0.321	N/A	N/A
	Griewank	6	2.400	0.200	N/A	N/A
HSAT	Ackley	11	0.013	N/A	6.48×10^{-6}	27
	Griewank	15	0.059	N/A	5.94×10^{-6}	24

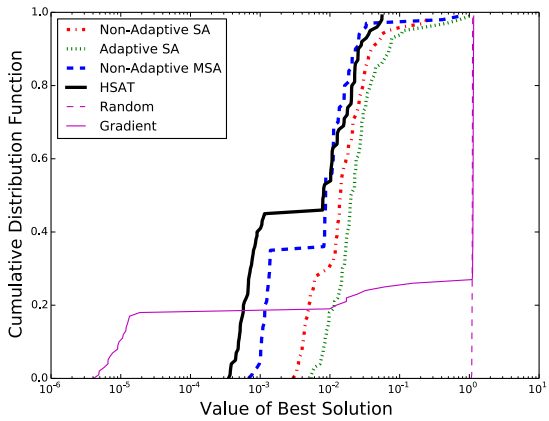
Every function was solved 100 times using each of the algorithms. A cumulative distribution function was then constructed for the final solutions, as shown in Figures 3(a) and 3(c). The cumulative distribution function shows the probability that a random variable (in this case, objective function value) will have a value equal to or less than a given value on the x-axis. The best solution returned by the algorithm was also tracked during optimization, as shown in Figures 3(b) and 3(d). Since algorithm parameters were tuned to improve performance for the given iteration limit, the average value of the best solution continues to improve throughout the allotted runtime (albeit slowly in some cases). Consequently, most algorithms achieve numerical convergence relatively late.



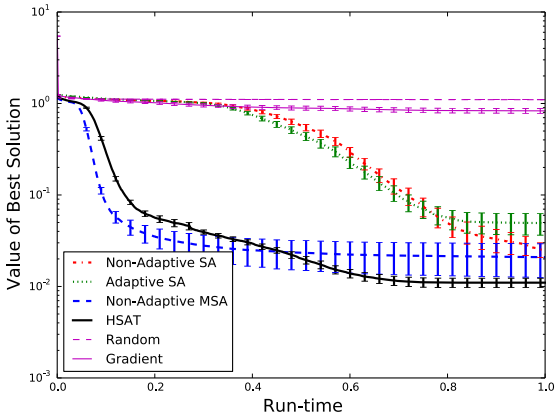
(a) Ackley function, cumulative distribution of final solutions.



(b) Ackley function, best solution over normalized run-time.



(c) Griewank function, cumulative distribution of final solutions.



(d) Griewank function, best solution over normalized run-time.

Figure 3. Comparison of optimization results (error bars show ±1 S.E.).

For the Ackley function, all SA-based algorithms outperform the gradient-based algorithm and the random search algorithm. However, there is a significant amount of differentiation between SA-based algorithms. The team-inspired HSAT algorithm returns the best final result, by nearly an order of magnitude. The HSAT algorithm also provides the most consistent results, as evidenced by the error bars in Figure 3(b). An examination of Figure 3(a) reveals the nature of the higher consistency. The other three SA-based algorithms occasionally produce final solutions that are local minima with objective function values on the order of 1.0, but the HSAT algorithm is capable of avoiding the local minima of the Ackley function. The algorithms that utilize adaptive annealing schedules (HSAT and Adaptive SA) are able to obtain the lowest objective function values. However, the addition of multiple, collaborating agents in the HSAT algorithm is necessary to also avoid local minima, thus ensuring high average performance. Similar results for the Griewank function are shown in Figures 3(c) and 3(d). In terms of mean performance, the HSAT algorithm out-performs all other algorithms (both SA and non-SA). In contrast to the results on the Ackley function, the highest-performing algorithms on the Griewank function are those that employ multiple agents. In approximately 80% of runs, the gradient-based algorithm performs only as well as random search. However, when a good initial starting location is chosen the For broader simple bounds, or for higher dimensionalities, the probability of randomly selecting a good starting point will decrease, making SA-based methods (and especially HSAT) more attractive.

The results on these benchmarking functions allow us to gain insight into the effect of the two features implemented in HSAT (independently-controlled adaptive annealing schedules, and probabilistic agent interaction). The Ackley function has numerous shallow local minima, but the global behaviour is readily apparent. The algorithms that employ adaptive annealing schedules (HSAT and adaptive SA) perform best, because they are capable of better responding to the global behaviour of the function. In contrast, the global behaviour of the Griewank function is obscured by the deep local minima. The algorithms that use multiple agents (HSAT and non-adaptive multi-agent SA) perform best, because they are capable of thoroughly searching a number of those local minima. Combining these features in HSAT ensures good performance on both functions.

5 CONCLUSIONS

This paper introduced the Heterogeneous Simulated Annealing Team (HSAT) algorithm, a multi-agent simulated annealing algorithm based upon two important properties that are also exhibited by human design teams. Every agent in the HSAT algorithm is given an independently adaptive annealing schedule, which allows the agent team to develop heterogeneous and asynchronous search strategies, resulting in behaviour similar to that of expert designers. Interaction between agents in HSAT is implemented in a way that probabilistically encourages agents to pursue better designs, while also allowing them to pursue worse designs. This mimics quality-informed interaction and allows for a pattern of divergence-to-convergence, both of which are observed in human teams.

The performance of the team-based HSAT algorithm was compared to other SA-based algorithms, a random search algorithm, and a gradient-based algorithm on two continuous functions. On both of the functions utilized in this work, HSAT demonstrated the best mean performance. The reason for this performance advantage appeared to depend on the function. For the Ackley function, high performance was likely a result of the adaptive annealing schedule. The high performance on the Griewank function appeared to be a result of multi-agent collaboration. This indicates that HSAT is not only capable of delivering high performance, but that this performance may also be robust across a variety of function topographies. Future work will adapt HSAT to discrete domain problems, and explore its applicability for the automation of complex design problems.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE125252 and the United States Air Force Office of Scientific Research through grant FA9550-12-1-0374.

REFERENCES

Azizi, N. and Zolfaghari, S., (2004) Adaptive temperature control for simulated annealing: a comparative study. Computers & Operations Research, Vol. 31, No. 14, pp. 2439–2451.

Bäck, T., 1996. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press.
Ball, L.J. and Ormerod, T.C., (1995) Structured and opportunistic processing in design: a critical discussion. International Journal of Human-Computer Studies, Vol. 43, No. 1, pp. 131–151.
Cagan, J., Dinar, M., Shah, J., Leifer, L., Linsey, L., Smith, S., and Vargas-Hernandez, N., (2013). Empirical Studies of Design Thinking: Past, Present, Future. ASME Design Theory and Methodology Conference, Portland OR, DETC2013–13302.
Cagan, J. and Kotovsky, K. (1997) Simulated Annealing and the generation of the objective function: a model of learning during problem solving. Computational Intelligence, Vol. 13, No. 4, pp. 534–581.
Campbell, M.I., Cagan, J., and Kotovsky, K. (1999) A-Design : An Agent-Based Approach to Conceptual Design in a Dynamic Environment. Research in Engineering Design, Vol. 11, No. 3, pp. 172–192.
Cho, H., Olivera, F., and Guikema, S.D. (2008) A derivation of the number of minima of the Griewank function. Applied Mathematics and Computation, Vol. 204, No. 2, pp.694–701.
Cross, N., 2004. Expertise in design: an overview. Design Studies, Vol. 25, No. 5, pp.427–441.
Dong, A., Hill, A.W., and Agogino, A.M. (2004) A Document Analysis Method for Characterizing Design Team Performance. Journal of Mechanical Design, 126(3), pp.378–385.
Egan, P.F., Cagan, J., Schunn, C., and LeDuc, P.R. (2014) Cognitive-based search strategies for complex bio-nanotechnology design derived through symbiotic human and agent-based approaches. ASME Design Theory and Methodology Conference, Buffalo NY, DETC2014–34714.
Finger, S. and Dixon, J.R. (1989) A Review of Research in Mechanical Engineering Design. Part 1: Descriptive, Prescriptive, and Computer-Based Models of Design Processes. Research in Engineering Design, Vol. 1, No. 1, pp.51–67.
Franklin, S. and Graesser, A. (1997) Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Intelligent Agents III Agent Theories, Architectures, and Languages. pp. 21–35.
Fu, K., Cagan, J., and Kotovsky, K. (2010) Design Team Convergence: The Influence of Example Solution Quality. Journal of Mechanical Design, Vol. 132, No. 11, p. 111005.
Goel, A.K., Vattam, S., Wiltgen, B., and Helms, M. (2012) Cognitive, collaborative, conceptual and creative — Four characteristics of the next generation of knowledge-based CAD systems: A study in biologically inspired design. Computer-Aided Design, Vol. 44, No. 10, pp. 879–900.
Granville, V., Krivanek, M., and Rasson, J.-P., (1994) Simulated Annealing: A Proof of Convergence. IEEE Transactions on pattern Analysis and Machine Intelligence, Vol. 16, No. 6, pp. 652–656.
Griewank, A.O., (1981) Generalized Descent for Global Optimization. Journal of Optimization Theory and Applications, Vol. 34, No. 1, pp. 11–39.
Hiroyasu, T., Miki, M., Ogura, S., Aoi, K., Yoshida, T., Okamoto, Y., and Dongarra, J. (2002) Energy Minimization of Protein Tertiary Structure by Parallel Simulated Annealing using Genetic Crossover SA and GA Parallel SA using Genetic. 2002 Genetic and Evolutionary Computation Conference. pp. 49–51.
Huang, M.D., Romeo, F., and Sangiovanni-Vincentelli, A.L., (1986) An efficient general cooling schedule for simulated annealing. IEEE International Conference on Computer-Aided Design. pp. 381–384.
Ingber, L. (1996) Adaptive simulated annealing (ASA): lessons learned. Controls and Cybernetics, Vol. 25, No. 1, pp. 33–54.
Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983) Optimization by simulated annealing. Science, Vol. 220, No. 4598, pp. 671–680.
Landry, L.H. and Cagan, J. (2011) Protocol-Based Multi-Agent Systems: Examining the Effect of Diversity, Dynamism, and Cooperation in Heuristic Optimization Approaches. Journal of Mechanical Design, Vol. 133, No. 2, p. 021001.
McComb, C., Cagan, J., and Kotovsky, K. (2014) Rolling with the punches: An examination of team performance in a design task subject to drastic changes. Design Studies. http://dx.doi.org/10.1016/j.destud.2014.10.001
Ohlidal, M. and Schwarz, J. (2004) Hybrid parallel simulated annealing using genetic operations. 10th International Conference on Soft Computing. pp. 89–94.
Papalambros, P.Y. and Wilde, D.J. (2000) Principles of Optimal Design, Cambridge University Press.
Triki, E., Collette, Y., and Siarry, P. (2005) A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. European Journal of Operational Research, Vol. 166, No. 1, pp. 77–92.
Tseng, I., Cagan, J., and Kotovsky, K. (2012) Concurrent Optimization of Computationally Learned Stylistic Form and Functional Goals. Journal of Mechanical Design, Vol. 134, No. 11, p. 111006.
Wood, M., Chen, P., Fu, K., Cagan, J., and Kotovsky, K. (2012) The Role of Design Team Interaction Structure on Individual and Shared Mental Models. In Design Computing and Cognition ’12. pp. 206–226.
Zhong, Y., Wang, L., Wang, C., and Zhang, H. (2012a) Multi-agent simulated annealing algorithm based on differential evolution. International Journal of Bio-Inspired Computation, Vol. 4, No. 4, pp. 217–228.
Zhong, Y., Ning, J., and Zhang, H. (2012b) Multi-agent simulated annealing algorithm based on particle swarm optimisation algorithm. International Journal of Computer Applications in Technology, Vol. 43, No. 4, pp. 335–342.