Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Image Compression and Recovery through Compressive Sampling and Particle Swarm

David B. Sturgill, Benjamin Van Ruitenbeek and Robert J. Marks II
Engineering and Computer Science
Baylor University
Waco, TX USA

***Abstract*—We present an application of particle swarm techniques to the problem of sparse signal recovery. Although a direct application of particle swarm is straightforward, specifics of the signal recovery problem can be incorporated into particle behavior in a way that substantially improves the quality of the recovered signal. With encouraging results for synthetic signals, we apply this technique to the problem of image compression, where typical image blocks can be expected to exhibit many very small elements under a transformation like the DCT. In this application, we observe that better results are obtained by first forcing image blocks to be sparse rather than compressively sampling blocks that are approximately sparse.**

***Index Terms*—Particle Swarm, Compressive Sampling, Image Compression**

I. Introduction

Compressive sampling can be an effective, space-efficient way to deal with signals that are known to have a sparse representation. Signal recovery presents computational challenges and several techniques have been developed to deal with this problem.

We present a particle swarm technique for sparse signal recovery. Particle swarm approaches are readily applicable to search problems in high-dimensional spaces, and a direct application of particle swarm to the signal recovery is fairly straightforward. Furthermore, constraints of the signal recovery problem can be incorporated into particle behavior to produce a system that is effective at recovering synthetic signals of a known sparsity. To evaluate performance on natural signals, we apply this technique to the problem of image compression.

A. Compressive Sampling

Compressive sampling [1], [2] is a method of encoding a sparse signal by taking a small number of non-adaptive linear measurements. A signal, $x \in \mathbb{R}^n$, can be considered k-sparse if its frequency-domain representation, Vx contains at most k non-zero elements for $k \ll n$ and some reasonable transformation, V. Under compressive sampling, the signal is represented by a short measurement vector Φx for some $m \times n$ sampling matrix, Φ. The number of measurements, m, is dependent on the sparsity, k, of the input signal.

A principal advantage of compressive sampling is that it requires a small number of samples to represent a signal and a small amount of storage to compute this compressed representation. However, *signal recovery*, the process of reconstructing the signal given the samples, can be an NP-hard problem.

B. Sparse Signal Recovery

Many approaches to the signal recovery problem have been developed, although they generally fall within two major categories: L_1-minimization and greedy techniques. The first of these depends on solving a convex relaxation of the L_0-minimization problem (finding the sparsest signal that produces the same measurements as the original). The convex relaxation of this problem can be solved using convex programming techniques; this method is known as *Basis Pursuit* [3], [4]. While it offers guarantees about the quality of the recovered signal, this technique can be computationally expensive.

Greedy algorithms reconstruct a signal by an iterative process that makes locally optimal decisions during each iteration. *Greedy Basis Pursuit* [5] uses greedy methods to enhance L_1-minimization. *Orthogonal Matching Pursuit* (OMP) [6] is a greedy algorithm which can provide a faster solution than the convex optimization approach. Other OMP-based algorithms such as stagewise OMP (StOMP) [7] and regularized OMP (ROMP) [8] improve the signal recovery of OMP. *Compressive Sampling Matching Pursuit* (CoSaMP) developed by Needell and Tropp [9] provides optimal signal recovery guarantees and efficient resource usage.

C. Particle Swarm

Particle Swarm Optimization (PSO), introduced by Eberhart and Kennedy [10], [11], is a search algorithm using a population of search agents called *particles*. The particles move through the search space with velocities which are dynamically adjusted according to their historical behavior [12], [13], [14]. PSO has found diverse and useful application in a number of disciplines [15], [16], including fractal image compression [17], [18] and lossless data compression [19]. PSO has also been shown to be effective in mutiobjective optimization [20], [21], [22], [23], [24].

The equations describing PSO require only two lines.

$$V_i \leftarrow V_i + c_1\, r_1()(P_i - X_i) + c_2\, r_2()(P_g - X_i) \tag{1}$$

and

$$X_i \leftarrow X_i + V_i, \tag{2}$$

where

- c_1 and c_2 are positive constants,
- $r_1()$ and $r_2()$ are random variables on $[0, 1]$,

- $X_1, \ldots X_D \in \mathbb{R}^d$ represent the d-dimensional position of each of D particles
- $V_1, \ldots V_D \in \mathbb{R}^d$ represent the velocity of each particle
- $P_1, \ldots P_D \in \mathbb{R}^d$ represents the best previous position of each particle as determined by a fitness function
- g represents the index of the particle with the best previous fitness.

At each iteration, particles move based on their velocity and their velocity is updated based on the location of the best known solutions.

II. Signal Recovery via Particle Swarm

It is straightforward to characterize sparse signal recovery as a particle swarm optimization problem. The space of possible signals can serve as a natural domain for particles to explore. The requirements that a solution agrees with the measurement, Φx, and that it exhibits the expected degree of sparsity can serve as components in the fitness function.

We investigate three progressively more sophisticated techniques for applying particle swarm to sparse signal recovery. The first technique attempts to directly recover the signal by using particles to explore the space of possible signals. A refinement of this technique constrains particles so that they always agree with the measurement, effectively reducing the dimensionality of the space explored. A final refinement extends the particle velocity update rule to pull particles toward solutions that exhibit the required sparsity.

A. Signal Recovery as Multiobjective Optimization

Our first technique attempts to directly recover x from the measurement by creating a population of particles distributed within the domain of x. We call this the *direct* technique. Particles begin with a random velocity and are drawn toward the locations of their own previous best and global best solution known. The fitness of a solution is based on two components, the degree to which the particle matches the measurement under Φ and the degree to which it exhibits the required level of sparsity under V. For particle i, the first of these components is measured as the squared Euclidean distance between ΦX_i and Φx. The second component is measured as the sum of the squared magnitudes of the $n-k$ elements of VX_i that are closest to zero. The fitness function is the sum of these two components, with the second component given 2.86 times the weight of the first. Particles with lower values are considered better.

We evaluate the effectiveness of this technique by measuring its performance on a collection of randomly generated signals. Each signal is drawn from $\mathbb{R}^{64}$ and exhibits sparsity between $k = 1$ and $k = 16$. We use the Discrete Cosine Transform as the V matrix, and signals are generated to so that k different random indices in Vx have values uniformly chosen from $[-1, 1]$. The number of samples, m is chosen as $3k + 5$ and Φ is simply the first m rows of a randomly generated orthonormal matrix.

We attempt to recover the original signal by sampling it with Φ and then using particle swarm to find an approximation of the original signal that matches the resulting m real-valued samples and exhibits the required level of sparsity under V. We have made some effort to tune this solution for the suite of synthetically generated signals. Particles are initialized with locations chosen uniformly randomly, $X_i \in [-1, 1]^n$. Particle velocities are initialized with uniform random coordinates $V_i \in [-0.5, 0.5]^n$. From iteration to iteration, particle velocity is updated as in (1), with $c_1 = 0.1$ and $c_2 = 0.005$. In an effort to focus particles on the interesting region of the search space, we damp particle motion from iteration to iteration. Just before applying (1), we reduce particle velocity by one percent. We use a population of 20 particles and permit the swarm to run for 10,000 iterations on each test case.

Fig 1 illustrates the performance of this approach on a collection of $1,600$ signals, 100 each at $k = 1 \ldots 16$. The Y axis reports average root mean squared error at each sparsity level.

B. Constrained Particle Swarm Search

The direct technique described in Section II-A attempts to find a location in the search space that satisfies two requirements, matching the measurement of the original signal under Φ and exhibiting the required sparsity under V. Satisfying the former requirement without the latter is a straightforward application of linear algebra. In fact, it is easy to describe the space of all locations x' such that $\Phi x' = \Phi x$. This leaves only the problem of finding points in this space that exhibit the required sparsity, the computationally hard aspect of the problem.

Our *constrained* technique represents an attempt to focus attention to the aspect of the problem that actually requires search. We confine particles to a space that maps to Φx under measurement. First, by solving an underdetermined system, we find a point S such that $\Phi S = \Phi x$. The set of points that map to Φx under measurement is simply $S + r$ for any r in the nullspace of Φ. Looking for an appropriate r in this space simplifies signal recovery by reducing the dimensionality of its search space.

In our second technique, instead of searching for the signal directly, we search for a vector r in the nullspace of Φ such that $V(S + r)$ is sufficiently sparse. Let the columns of matrix N_Φ be an an orthonormal basis for the nullspace of Φ. Particles explore the space of feasible solutions by looking for coefficients for these columns. Particle fitness is assessed by measuring how close the particle is to the required sparsity. We measure the fitness of particle i as the sum of the squared magnitudes of the $n-k$ smallest elements of $VS + VN_\Phi X_i$.

Fig 1 plots the average root mean squared error for constrained particle swarm recovery in comparison to the direct technique described in Section II-A. The influence of the local and global best solutions is the same as before, and the particle initialization is analogous. We initialize particle i by first generating a random location in $X_i' \in [-1, 1]^n$ and a velocity in $V_i' \in [-0.5, 0.5]^n$. We then project this to the particle's $n - m$ dimensional domain by solving the overdetermined systems:

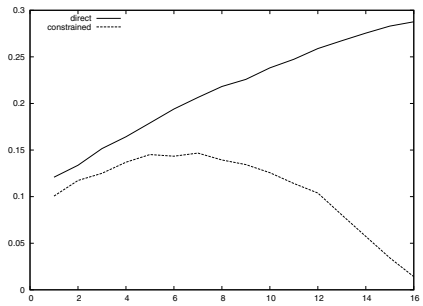

Fig. 1. Comparison of root mean squared error for direct and constrained particle swarm techniques. The X axis indicates the sparsity, k, and the Y axis gives average error across 100 signals at each sparsity level.

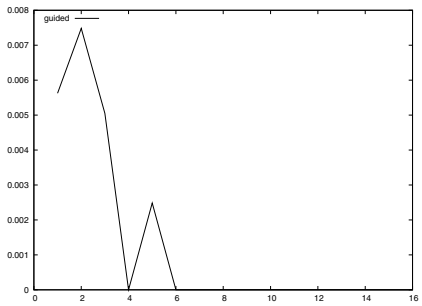

Fig. 2. Root mean squared error for the guided swarm technique. This graph plots average error against k for the same $1,600$ signals described in Fig 1.

$$VS + VN_\Phi X_i = X_i'$$
$$VN_\Phi V_i = V_i'$$

From Fig 1, we see that the constrained technique performs slightly better than the direct technique for small values of k. As k becomes larger, the difference becomes much more pronounced. This is to be expected, since more measurement values further constrain the search. For $k = 16$, the direct technique is still searching in $\mathbb{R}^{64}$, while the constrained technique is searching in $\mathbb{R}^{11}$.

C. Guided Particle Swarm Search

Under the direct and constrained techniques, the fitness function serves as an indication of where the most promising regions of the search space are. As approximate solutions are found, particles are pulled toward them when their velocities are updated via (1). Instead of relying on previous discoveries as the only guide to particles, it's possible to use the linear mapping between the particle and its image under V to help steer particles to the required sparsity. Our *guided* technique is an attempt to exploit this.

Let U_i be the image of particle i under V.

$$U_i \quad = \quad VS + VN_\Phi X_i$$

Vector U_i describes particle i in the frequency domain. In this representation, it's easy to capture the smallest change to particle i sufficient to achieve the required sparsity. We just need zero out the $n-k$ smallest magnitudes. We define U_i' as follows, so that so that $U_i + U_i'$ is k-sparse:

$$U_{ij}' = \begin{cases} 0 & \text{if } U_{ij} \text{ is among the largest } k \text{ magnitudes in } U_i \\ -U_{ij} & \text{otherwise} \end{cases}$$

The vector U_i' is used to pull particles toward a nearby k-sparse location. During search, particle velocity is updated based on three contributions, two that are typical of particle swarm approaches and one that is computed from U_i' and is specific to sparse signal recovery. Of course, the n-dimensional, frequency-domain U_i' can't be applied to the lower-dimensional particle velocity directly. Instead, we use least-squared techniques to compute a corresponding vector Z_i that helps to move the particle in a direction that promotes sparsity in its frequency-domain representation. At each iteration, particle velocity is updated as follows:

$$\begin{aligned} V_i \quad = \quad & V_i + \\ & c_1\ r_1()(p_i - x_i) + \\ & c_2\ r_2()(p_g - x_i) - \\ & c_3\ r_3()Z_i \end{aligned}$$

where c_3 is a positive constant, $r_3()$ is a random variable on $[0, 1]$ and Z_i is a least-squares solution to the overdetermined system:

$$VN_\Phi Z_i = U_i'$$

Fig 2 presents the root mean squared error for the guided technique on the same $1,600$ signals used in Fig 1. This system does a much better job of recovering the original signal, reproducing it almost exactly for all test cases with k larger than 6. Even for the test cases with smaller values of k, average error is much better than for the direct or constrained techniques. For these cases, the deviation from zero represents the contribution of small number of signals where the particle swarm does not find the correct pattern of non-zero elements in the signal's sparse representation.

III. Application to Image Compression

The results presented in Fig 2 are encouraging, but they describe performance on synthetic data. To evaluate performance on real-world signals, we have adapted the guided technique to an image compression application. In fact, many of the parameters used in previous experiments were chosen with this application in mind.

A. Image Block Encoding

We use compressive sampling to encode 8-bit grayscale images. The image is partitioned into 8×8 blocks of pixels, and each block is encoded independently as a signal in $\mathbb{R}^{64}$ occurring in row-major order in the block. Pixel intensity is mapped from the typical $[0, 255]$ range given in the input to the $[-1, 1]$ range common to our previous experiments.

Transformation V is a two-dimensional DCT. For natural images, we can expect typical blocks to be approximately

sparse under this transformation; a DCT-transformed block is likely to have many values that are very close to zero. We measure the sparsity of an image block as the number of values with magnitude greater than 10^{-1}. We expect this measure of sparsity to vary considerably across the image, so we deal with sparsity adaptively. We compressively sample each block just as in previous experiments, setting k based on the block's measured sparsity under the DCT and setting the number of samples, m, to be $3k + 5$.

To measure the actual storage cost for the compressively sampled image, we encode the measurement for each block using a linear quantization. Intensity values in the $[-1, 1]$ range are transformed via a Φ matrix with a number of rows determined by block sparsity. The resulting values are scaled by a factor of 64 and then rounded to the nearest integer. During image recovery, midpoints of the quantization intervals are used for each measurement value.

With an 8×8 block size and a scaling factor of 64, a measurement value could require up to 10 bits of storage. However, we have observed that most blocks don't actually exhibit the extremes in the range of possible measurements. In an effort to reduce storage cost, we encode the measurement for each block using the fewest bits sufficient to represent all values that actually occur. We prefix the encoding of each block with a four-bit field indicating how many bits the block's measurement values require.

In an effort to further reduce encoding size, the DC component of the transformed image block is given special treatment. We encode this value as a separate 8-bit quantity for each block. Before compressive sampling, the DC component is set to zero and the image block is reconstructed via the inverse DCT. This has the effect of centering the average block intensity on zero, helping to reduce the magnitudes in the measurement and often reducing the number of bits required to encode a measurement value.

B. Image Block Sparsity

Since an image can only be expected to be approximately sparse, there is some uncertainty about how image blocks should be compressively sampled and recovered. The most straightforward approach might be to compressively sample the approximately sparse blocks from the original image. Alternatively, we could force approximately k-sparse image blocks to be exactly k-sparse by zeroing out elements of the DCT-transformed image block with magnitude less than 10^{-1}. We could then apply the inverse DCT to reconstruct an image block similar to the original one and then compressively sample that. We say the former approach is sampling an image block with *approximate* sparsity and the latter is sampling one with *exact* sparsity. During signal recovery, we might expect approximate sparsity to produce a better match for the original image because samples are taken directly from the original. Under exact sparsity, some information is discarded before the measurement matrix is applied. Of course, if we sample blocks with exact sparsity, it is at least possible to find a k-sparse signal that matches the block's measurement.

TABLE I
IMAGE COMPRESSION PERFORMANCE ON 512×521 GRAYSCALE IMAGES.

Image	Sparsity	Encoding Size (bits)	Recovery Time (sec)	RMS Error
4.2.04	Approx	566068	5933.9	5.50
4.2.04	Exact	554231	5824.8	3.87
4.2.07	Approx	579512	6150.4	6.24
4.2.07	Exact	564591	6035.5	4.25
5.2.08	Approx	740736	6547.8	5.46
5.2.08	Exact	729794	6462.9	3.82
5.2.10	Approx	1485349	3468.7	5.17
5.2.10	Exact	1472737	3381.7	4.79
boat.512	Approx	830779	5960.9	6.21
boat.512	Exact	816821	5838.4	4.27
elaine.512	Approx	632381	7627.3	7.78
elaine.512	Exact	606386	7519.7	4.89

C. Experimental Evaluation

We have applied this compression technique to six images taken from the USC-SIPI Image Database [25]. All images are 512×512 pixels, with some converted to grayscale for processing here.

Table I reports the encoding size in bits, the total time to recover the entire image and the root mean squared error across the entire image. These values are reported for sampling of the original, approximately sparse image and the exactly sparse image with small Discrete Cosine Transformed elements discarded. In both cases, the reported RMS error compares the recovered image with the original image, with intensity values in the $[0, 255]$ range.

Results reported here uniformly favor recovery via exact sparsity. It exhibits a slight advantage in encoding size and processing time to recover the image. These results are, perhaps, to be expected since information was discarded to create the exactly sparse image, and it is possible for the particle swarm technique to settle in on a k-sparse solution that agrees with the measurement. The advantage in error is more surprising since the exact sparsity technique recovers a better match for the original image with less information from the image.

Fig 3 and 4 compare the original image with the image recovered via compressive sampling. The detail shown in Fig 5 helps to illustrate where this technique fails in reconstructing the image. Often, image block recovery is able to find the correct pattern of non-zero elements in the block's sparse representation. These blocks closely match the original. For a few blocks, the system does not succeed in finding the right pattern and the recovered block exhibits artifacts typical of DCT-encoded signals. This can help to explain the apparent advantage of recovery via exact sparsity. For the exact sparsity technique, some information is discarded at the outset, but it is at least possible to find a pattern of k non-zero elements that matches the measurement. Under approximate sparsity, it is not possible to find such a pattern. With sparsity as the only indicator of solution quality available to the guided system, there may not be enough to distinguish the neighborhood of

Original

Recovered via Approximate Sparsity

Fig. 3. Comparison of the original 4.2.04 image and an image recovered via compressively sampled, approximately k-sparse blocks

Original

Recovered via Exact Sparsity

Fig. 4. Comparison of the original 4.2.04 image and an image recovered from via compressively sampled, exactly k-sparse blocks

the best approximately sparse solution from inferior ones.

The results reported here do not compare well with standard, lossy image compression techniques. At quality setting 80, JPEG produces both smaller encoding sizes and lower error than virtually all the the results reported in Table I. For example, while we achieve an error of 4.89 using 606386 bits to encode the elaine.512 image, at quality 80, JPEG exhibits an error of 4.71 using only 343104 bits.

IV. Conclusions

We demonstrate that particle swarm techniques can be effectively employed in recovering compressively sampled signals. Although naïve approaches do not readily converge to a close approximation of the original signal, constraining and guiding particle behavior based on specifics of the signal recovery problem can improve performance considerably. On a collection of 1,600 synthetic signals from $\mathbb{R}^n$, we are able to recover all signals with sparsity $k = 6$ or greater with $3k + 5$ floating point measurement values.

An application of our best technique to natural images helps to demonstrate some practical difficulties. Even with some effort to encode images parsimoniously, compressed image size and the quality of the recovered image are not competitive with common compression techniques like JPEG. Likewise, the computational overhead associated with image recovery makes this particular technique impractical as an image compression scheme.

As an example of a sparse signal, the Discrete Cosine Transform of an image block may feature many elements that are very small but not exactly zero. The right way to deal with this approximate sparsity presents another challenge. For our particle swarm approach, we find that discarding some

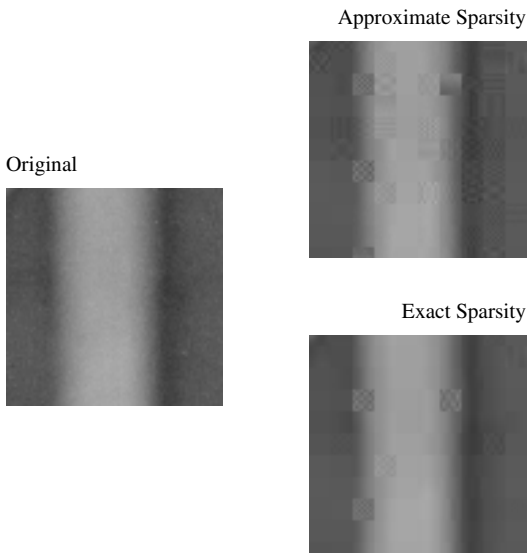

Fig. 5. Detail from the left edge of the 4.2.04 image, comparing the original and images recovered via approximate and exact sparsity

information at the outset by forcing very small elements of the transformed image block to zero yields an overall better reproduction of the original image.

REFERENCES

[1] D. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[2] E. Candes and M. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, March 2008.

[3] S. Chen and D. Donoho, "Basis pursuit," vol. 1, Oct-2 Nov 1994, pp. 41–44 vol.1.

[4] S. S. B. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal of Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1999.

[5] P. Huggins and S. Zucker, "Greedy basis pursuit," *Signal Processing, IEEE Transactions on*, vol. 55, no. 7, pp. 3760–3772, July 2007.

[6] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *Information Theory, IEEE Transactions on*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[7] D. L. Donoho, Y. Tsaig, I. Drori, and J. luc Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," Tech. Rep., 2006.

[8] D. Needell and R. Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," Mar. 15 2007, comment: This is the final version of the paper, including referee suggestions.

[9] J. A. Tropp and D. Needell, "CosaMP: Iterative signal recovery from incomplete and inaccurate samples," *CoRR*, vol. abs/0803.2392, 2008, informal publication.

[10] R. Eberhart and J. Kennedy, "Particle swarm optimization: developments, applications and resources," in *A new optimizer using particle swarm theory*, 1995, pp. 39–43.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[12] R. Eberhart, P. Simpson, and R. Dobbins, *Computational intelligence PC tools*. San Diego, CA, USA: Academic Press Professional, Inc., 1996.

[13] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the Congress on Evolutionary Computation*, 2001.

[14] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*, ser. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, March 2001.

[15] R. Eberhart and Y. Shi, *Computational Intelligence: Concepts to Implementations*. Morgan Kaufmann, 2007.

[16] R. Poli, J. Kennedy, T. Blackwell, and A. Freitas, *Particle Swarms: The Second Decade*. Hindawi Publishing Corp, 2008.

[17] H.-M. Feng, C.-Y. Chen, and F. Ye, "Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression," *Expert Syst. Appl.*, vol. 32, no. 1, pp. 213–222, 2007.

[18] J.-G. Hsieh, J.-H. Jeng, and C.-C. Tseng, "Study on huber fractal image compression," *IEEE Transactions on Image Processing*, in press.

[19] D. Shuai, P. Zhang, and B. Zhang, "Particle algorithm for lossless data compression," in *IEEE International Conference on Systems, Man and Cybernetics*, Oct 2006, pp. 3766–3771.

[20] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, June 2004.

[21] S. Ho, S. Yang, G. Ni, E. Lo, and H. Wong, "A particle swarm optimization-based method for multiobjective design optimizations," *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1756–1759, May 2005.

[22] D. Liu, K. Tan, C. Goh, and W. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 1, pp. 42–50, Feb 2007.

[23] E. Ochlak and B. Forouraghi, "A particle swarm algorithm for multiobjective design optimization," *Tools with Artificial Intelligence, IEEE International Conference on*, vol. 0, pp. 765–772, 2006.

[24] C. Tan, C. Goh, K. Tan, and A. Tay, "A cooperative coevolutionary algorithm for multiobjective particle swarm optimization," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 3180–3186.

[25] A. Weber, "The usc-sipi image database," http://sipi.usc.edu/database/.