Operations Research Letters 38 (2010) 550–555

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Speeding up COMPASS for high-dimensional discrete optimization via simulation

L. Jeff Hong [a,*], Barry L. Nelson [b], Jie Xu [b]

[a] *Department of Industrial Engineering and Logistics Management, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China*

[b] *Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208, United States*

ARTICLE INFO

Article history:
Received 29 April 2010
Accepted 30 August 2010
Available online 21 September 2010

Keywords:
Discrete optimization via simulation
COMPASS algorithm
Sampling

ABSTRACT

The convergent optimization via most promising area stochastic search (COMPASS) algorithm is a locally convergent random search algorithm for solving discrete optimization via simulation problems. COMPASS has drawn a significant amount of attention since its introduction. While the asymptotic convergence of COMPASS does not depend on the problem dimension, the finite-time performance of the algorithm often deteriorates as the dimension increases. In this paper, we investigate the reasons for this deterioration and propose a simple change to the solution-sampling scheme that significantly speeds up COMPASS for high-dimensional problems without affecting its convergence guarantee.

1. Introduction

The COMPASS algorithm was first proposed by Hong and Nelson [1] to solve discrete optimization via simulation (DOvS) problems. In every iteration of the algorithm, a most promising area (MPA) is constructed around the current sample-best solution by randomly choosing some new solutions from the MPA, evaluating (simulating) all of the chosen solutions (in the current and previous iterations) based on a simulation–allocation rule, and then selecting the solution with the sample-best performance. Hong and Nelson [1] showed that, under some mild conditions, the algorithm converges to the set of locally optimal solutions with probability 1, as the amount of simulation effort goes to infinity for DOvS problems with a finite or a countably infinite number of solutions. Since its invention, COMPASS has drawn a significant amount of attention and has been applied to solve problems from various areas, including project management [3] and supply chain design [4].

Hong and Nelson [2] later developed a framework for locally convergent random search algorithms and showed that COMPASS is a special case of the framework. The framework allowed them to prove that only a subset of all chosen solutions – as opposed to the accumulation of all solutions ever chosen – needs to be simulated in each iteration to ensure local convergence; this insight speeded up the algorithm significantly. Xu et al. [7] embedded this modified COMPASS algorithm in software called Industrial Strength COMPASS (ISC). ISC has three phases. In the first phase, it conducts a global search and identifies several regions where there may exist good locally optimal solutions. In the second phase, ISC applies COMPASS to each of the identified regions to search for a locally optimal solution. In the third phase, ISC selects the best of the locally optimal solutions it has discovered and estimates the selected solution's performance to a desired level of precision. Xu et al. [7] compared ISC to OptQuest (OptTek Systems, Inc.), a well-known commercial solver that is integrated into 13 simulation products (http://www.opttek.com/simulation.html). They reported that ISC has competitive or superior performance compared to OptQuest for problems with small to moderate dimensions (5–10). For high-dimensional problems (e.g., 15 or 20), however, ISC was significantly slower than OptQuest in identifying good solutions and required a lot more (often 20 or 30 times more) computational overhead, even though it eventually finds solutions that are as good or better than OptQuest does.

To help COMPASS demonstrate real "industrial strength", it is important to investigate why COMPASS slows down for high-dimensional problems and to provide modifications that alleviate the problem. In this note, we report on such an investigation whose primary finding is that the deterioration of COMPASS is caused by the uniform solution-sampling scheme, used to choose solutions from the MPA. Both empirical and analytical evidence shows that uniform sampling is not efficient for finding better solutions. To alleviate this problem, we suggest a coordinate sampling scheme which randomly chooses solutions that differ in only one coordinate from the current sample-best solution. Both empirical and analytical evidence shows that this is a much more efficient sampling scheme, especially when the dimension of the problem is high. For a 15-dimensional quadratic problem that we tested, the COMPASS algorithm using coordinate sampling *reduces the amount of simulation effort by at least* 10 *times and reduces the computational overhead by at least* 1000 *times* compared to using uniform sampling. Because uniform sampling is widely used in

* Corresponding author. Tel.: +852 23587096.
E-mail address: hongl@ust.hk (L.J. Hong).

0167-6377/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2010.09.003

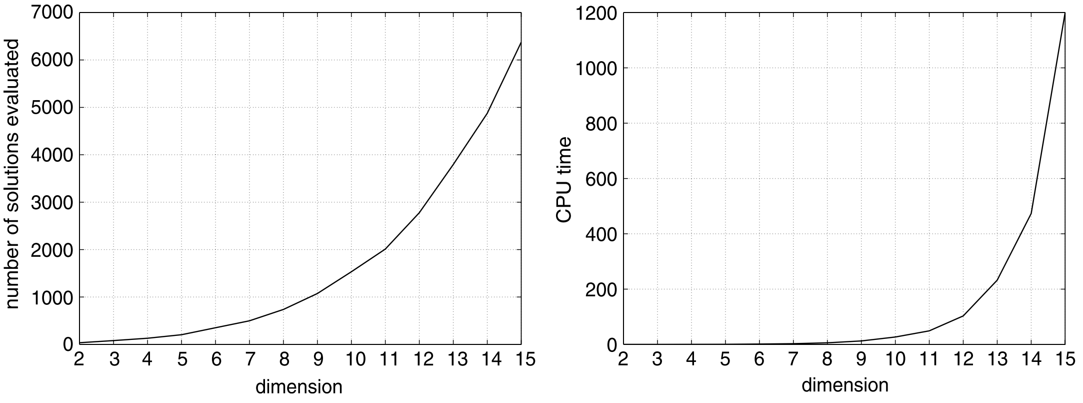

Fig. 1. The performance deterioration of COMPASS as the dimension increases in terms of the number of solutions evaluated (left) and the CPU time (right).

many random search algorithms, due to its simplicity and intuitive appeal, we believe that the knowledge we obtained for COMPASS may also be valuable in speeding up other algorithms for high-dimensional problems.

The remainder of this paper is organized as follows. In Section 2 we investigate two plausible reasons as to why COMPASS may slow down for high-dimensional problems. In Section 3, we analyze the coordinate sampling scheme. Numerical results are reported in Section 4, followed by conclusions in the Appendix.

2. Uncovering the reason of deterioration

We are interested in solving the following problem:

$$\text{minimize } g(\mathbf{x}) = \mathrm{E}[G(\mathbf{x})] \quad \text{subject to } \mathbf{x} \in \Theta = \Phi \cap \mathcal{Z}^d \tag{1}$$

where $\mathbf{x}$ is a vector of d integer-ordered decision variables in a convex feasible region $\Phi \subset \Re^d$, possibly defined by a set of constraints. The random variable $G(\mathbf{x})$ typically has no closed form, but can be observed through simulation experiments at $\mathbf{x}$. Without loss of generality, we assume that Φ is a compact set and, therefore, $|\Theta| < \infty$. By using the technique of bounding boxes of Hong and Nelson [1], the results of this paper can be extended easily to the problems where Φ is unbounded.

Any iterative random search algorithm that solves Problem (1) has two main components in each iteration: a solution-sampling scheme that (randomly) chooses one or more feasible solutions $\mathbf{x}$, and an estimation scheme that decides how many simulation observations to obtain for all chosen solutions [2]. Let $\mathcal{V}_k$ denote the set of all chosen solutions through iteration k, and let $\hat{\mathbf{x}}_k^*$ denote the sample-best solution in iteration k. In each iteration (say iteration k), COMPASS samples m solutions uniformly from the most promising area (MPA), defined as

$$\mathcal{C}_k = \left\{ \mathbf{x} \in \Theta : (\hat{\mathbf{x}}_k^* - \mathbf{y})^T \left(\mathbf{x} - \frac{\hat{\mathbf{x}}_k^* + \mathbf{y}}{2} \right) \geq 0, \right.$$
$$\left. \forall \mathbf{y} \in \mathcal{V}_k \text{ and } \mathbf{y} \neq \hat{\mathbf{x}}_k^* \right\}. \tag{2}$$

Note that the inequality in Eq. (2) defines a cut that passes through the mid-point of the line segment connecting $\mathbf{y}$ and $\mathbf{x}_k^*$ and, therefore, $\mathcal{C}_k$ includes all feasible solutions that are at least as close to $\hat{\mathbf{x}}_k^*$ as to other solutions in $\mathcal{V}_k$. Next the algorithm updates $\mathcal{V}_k$ to include all newly sampled solutions and estimates $g(\mathbf{x})$ for all solutions $\mathbf{x} \in \mathcal{V}_k$ based on a simulation–allocation rule. At the end of the iteration, the algorithm selects the solution with the lowest cumulative sample mean from all solutions in $\mathcal{V}_k$ as $\hat{\mathbf{x}}_{k+1}^*$. Xu et al. [7] extended the definition of the MPA to allow the cut to be placed adaptively at any point on the line segment. However, this extension does not appear to improve the performance of COMPASS for higher-dimensional problems.

Through a comprehensive empirical study, Xu et al. [7] showed that the ISC algorithm, which uses COMPASS as its core, is efficient in solving DOvS problems if the dimension of the problems is not high. However, the performance of ISC deteriorates significantly when the dimension of the problem increases. Because this deterioration also happens even when the stochastic noise level of the problem is low, we investigated the solution-sampling scheme of COMPASS as a possible cause.

2.1. A simple numerical example

Consider the following problem:

$$\text{minimize } g(\mathbf{x}) = \mathbf{x}^T\mathbf{x}, \quad \text{subject to } \mathbf{x} \in [-100, 100]^d. \tag{3}$$

Problem (3) is a very simple quadratic problem and its optimal solution is located at $\mathbf{x}^* = \mathbf{0}$. The problem has no stochastic noise. Therefore, we only need to evaluate a newly chosen solution once to know its objective value. This allows us to eliminate the effect of the estimation scheme and to focus solely on the solution-sampling scheme used in COMPASS.

We first apply COMPASS on Problem (3) by varying the dimension from 2 to 15. Because COMPASS is a random search algorithm, we run it 10 times for each dimension and report the average performance. In all the numerical experiments reported in this note, we set $m = 5$, i.e., we sample five solutions from the MPA in each iteration. To understand the level of deterioration as the dimension of the problem increases, we examine two performance measures. The first is the number of solutions that the algorithm evaluates until it first hits the optimal solution $\mathbf{x}^*$, and the second is the CPU time that the algorithm spends until it first hits $\mathbf{x}^*$. The first measure quantifies the amount of simulation effort required by COMPASS, and the second measure quantifies the amount of computational overhead in running COMPASS (because the CPU time for evaluating $g(\mathbf{x})$ is negligible in this example). We plot the two performance measures with respect to the dimension of the problem in the two panels of Fig. 1.

From the left panel of Fig. 1, we can see that the number of solutions required to first hit $\mathbf{x}^*$ increases from an average of 204.4 for the 5-dimensional problem, to 1532.7 for the 10-dimensional problem, and to 6575.4 for the 15-dimensional problem. This implies that COMPASS needs significantly more simulation effort to solve problems with higher dimension. From the right panel of Fig. 1, we can see that the CPU time increases even faster. It increases from 0.49 s for the 5-dimensional problem to 26.8 s for the 10-dimensional problem, and to 1200.5 s (about 20 min) for the 15-dimensional problem. This confirms that the computational overhead of COMPASS grows extremely fast as the dimension increases, which was also observed by Xu et al. [7].

2.2. Investigating the solution-sampling scheme

Note that the solution-sampling scheme of COMPASS includes two components: the first is the construction of the MPA and the second is the uniform sampling within the MPA. To understand the deterioration of COMPASS for high-dimensional problems, we first investigate the quality of the MPA. The MPA in Eq. (2) is constructed such that it "includes all feasible solutions that are at least as close to $\hat{\mathbf{x}}_k^*$ as to other solutions in $\mathcal{V}_k$" [1]. The idea behind this construction is that the (locally) optimal solution is likely to be closer to the current sample-best solution than to all other visited solutions. Therefore, a good way to test the quality of the MPA is to check how frequently the optimal solution is actually in the MPA constructed in each iteration. We calculate this performance measure for Problem (3) by varying the dimension of the problem from 2 to 15 and running COMPASS 10 times for each dimension. We find that *the optimal solution* $\mathbf{x}^*$ *is always in the MPA for all iterations and all dimensions*. This experiment suggests that at least for this problem the MPA is of good quality, and we should analyze the other component of the solution-sampling scheme to uncover the reason for the deterioration.

Uniform sampling is popular in random search algorithms because it gives every solution the same non-zero probability of being sampled so that the optimal solution will not be missed infinitely often if it is in the MPA. The nested partitions algorithms of Shi and Ólafsson [6] and Pichitlamken and Nelson [5], as well as the COMPASS algorithm, use uniform sampling. Also, uniform sampling is easy to implement. However, the efficiency of uniform sampling is rarely studied in the literature.

To analyze this efficiency, we record the percentage of the iterations that COMPASS samples a better solution (i.e., a solution that is better than the current sample-best solution) until it first hits $\mathbf{x}^*$. If the percentage is high, then the algorithm is likely to improve on every iteration. Otherwise, it is likely to stay where it is and waste effort in evaluating inferior solutions. In Fig. 2, we plot the percentages with respect to the dimensions of Problem (3). From the plot, we can see that the percentage goes down very quickly. It reduces from 36.1% for the 5-dimensional problem, to 11.1% for the 10-dimensional problem, and to only 4.01% for the 15-dimensional problem. Therefore, when the dimension of the problem is 15, even though five solutions are sampled at every iteration, *better solutions are only found in one out of every* 25 *iterations on average*. This means that the algorithm spends too much effort sampling and evaluating inferior solutions when the dimension is high. Furthermore, when the number of sampled solutions increases, the MPA of Problem (2) becomes more complicated and uniform sampling in the MPA becomes more time-consuming to conduct. This explains the dramatically increasing resource consumption that we have seen in the panels of Fig. 1. Therefore, we conclude that the uniform sampling scheme largely causes the deterioration of COMPASS for high-dimensional problems.

2.3. Understanding uniform sampling

To understand the behavior of uniform sampling for high-dimensional problems, we consider a simple model where the MPA is a hypercube $[-1, 1]^d$, the optimal solution is located at $\mathbf{0}$, and the quality of the solution depends on its Euclidean distance to $\mathbf{0}$. Suppose that the current sample-best solution is $\hat{\mathbf{x}}^*$ and we relax the integrality requirement. Then all solutions in the d-dimensional hypersphere $\mathcal{S}(d) = \{\mathbf{x} : \|\mathbf{x}\| < \|\hat{\mathbf{x}}^*\|\}$ have better quality than the current sample-best solution $\hat{\mathbf{x}}^*$, where $\|\mathbf{x}\|$ denotes the Euclidean norm of $\mathbf{x}$ which is the distance between $\mathbf{x}$ and $\mathbf{0}$ (see Fig. 3 for a 2-dimensional illustration). Let $p(d)$ denote the probability of sampling a solution that is better than $\hat{\mathbf{x}}^*$ in the MPA. For uniform sampling this is

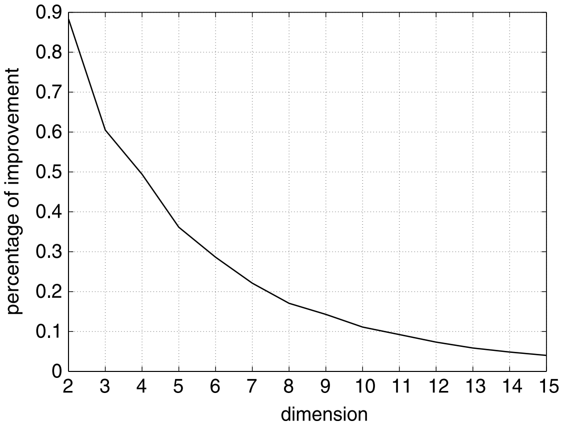

Fig. 2. The percentage of iterations that COMPASS finds a better solution as the problem dimension increases.

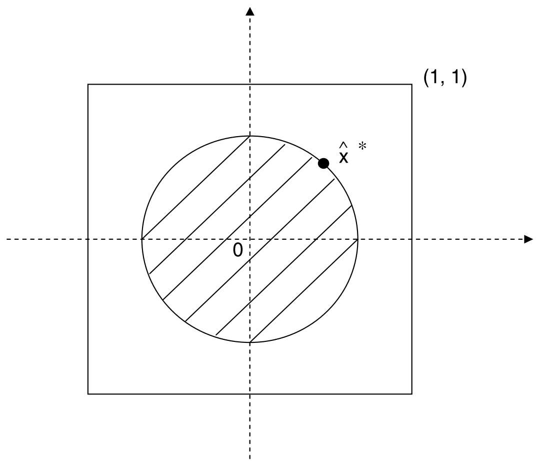

Fig. 3. The model of uniform sampling in the MPA.

$$p(d) = \frac{V(\mathcal{S}(d) \cap [-1, 1]^d)}{2^d}$$

where $V(A)$ denotes the volume of a d-dimensional set A. If we sample m solutions independently from the MPA, then the probability that at least one better solution will be sampled is given by

$$P(d) = 1 - [1 - p(d)]^m$$

where $m = 5$ in our experiments.

When we construct the MPA by using Eq. (2), the current sample-best solution is often significantly away from the boundary of the MPA. Suppose that $\|\hat{\mathbf{x}}^*\| = r \leq 1$, which means that $\hat{\mathbf{x}}^*$ is inside the inscribed hypersphere of the hyperbox. Then, by Yoshioka [8],

$$p(d) = \frac{V(\mathcal{S}(d))}{2^d} = \frac{\pi^{d/2} r^d}{2^d \Gamma(\frac{d}{2} + 1)} \leq \frac{\pi^{d/2}}{2^d \Gamma(\frac{d}{2} + 1)}. \quad (4)$$

This upper bound on $p(d)$ decreases very rapidly as d increases. This provides intuition for the phenomenon that we observe in Fig. 2.

Although this model is a simplification, it explains why uniform sampling in the MPA does not work well as the dimension of a DOvS problem increases. The set of better solutions becomes a smaller and smaller subset of the MPA. Hence, it becomes more and more difficult for uniform sampling to identify better solutions in the MPA. To speed up the performance of COMPASS for high-dimensional problems we will have to look for other solution-sampling schemes that are more likely to identify better solutions.

3. Speeding up COMPASS using coordinate sampling

From the analysis in Section 2, it is clear that the uniform sampling scheme is a primary reason for the deterioration of COMPASS for high-dimensional problems. To alleviate the difficulty, we propose a simple solution-sampling scheme, called *coordinate sampling*. It works as follows on iteration k of COMPASS.

Coordinate Sampling

Step 1. Uniformly sample an integer i from 1 to d.

Step 2. Determine the lower bound x_{ld} and upper bound x_{ud} such that $[\hat{\mathbf{x}}_k^* + x_{ld}\mathbf{e}_i, \hat{\mathbf{x}}_k^* + x_{ud}\mathbf{e}_i] \subset \mathcal{C}_k$, $\hat{\mathbf{x}}_k^* + (x_{ld} - 1)\mathbf{e}_i \notin \mathcal{C}_k$ and $\hat{\mathbf{x}}_k^* + (x_{ud} + 1)\mathbf{e}_i \notin \mathcal{C}_k$, where $\mathbf{e}_i$ is the ith column of a $d \times d$ identity matrix, $\mathcal{C}_k$ is the MPA in iteration k and $\hat{\mathbf{x}}_k^*$ is the sample-best solution in iteration k.

Step 3. Sample an integer solution uniformly from $[\hat{\mathbf{x}}_k^* + x_{ld}\mathbf{e}_i, \hat{\mathbf{x}}_k^* + x_{ud}\mathbf{e}_i]$.

Note that solutions sampled in this way will differ from $\hat{\mathbf{x}}_k^*$ in only one coordinate. Furthermore, it is worthwhile to observe that coordinate sampling is the first step of the revised mixed-D algorithm that is used in COMPASS to conduct uniform sampling. Therefore, the computational overhead of coordinate sampling is significantly smaller than that of uniform sampling which requires many steps. In the remainder of this section we explain why we choose coordinate sampling, how it performs, and whether it preserves the convergence properties of the original COMPASS algorithm.

3.1. Understanding coordinate sampling

To understand the advantages of coordinate sampling, we use the same model developed in Section 2.3. Given the model, the set of better solutions is the hypersphere $\mathcal{S}(d)$ (see Fig. 4 for a 2-dimensional illustration). Note that conditional on coordinate direction i being chosen, the probability of sampling a better solution is $|\hat{x}_i^*|$. As all coordinate directions are sampled with an equal probability $1/d$, the coordinate sampling gives

$$p(d) = \frac{1}{d}\sum_{i=1}^{d} |\hat{x}_i^*|$$

where $\hat{\mathbf{x}}^* = (\hat{x}_1^*, \ldots, \hat{x}_d^*)^T$. Suppose that $\|\hat{\mathbf{x}}_k^*\| = r$. Then it is easy to show that

$$\frac{r}{d} \le p(d) \le \frac{r}{\sqrt{d}},$$

where both bounds decrease at a much slower rate than the one for uniform sampling (see Eq. (4)). Although the model is a simplification, it suggests an advantage of coordinate sampling for high-dimensional problems.

To test the validity of our analysis, we implement coordinate sampling in COMPASS, apply it to solve Problem (3), and compare it to the performance of the original COMPASS that uses uniform sampling. We use the same settings as the ones in Section 2 and report the results in Table 1, where "# evaluated solutions", "CPU time" and "% improvement" mean the average number of solutions evaluated by COMPASS, the average CPU time, and the percentage of iterations that the algorithm finds at least one better solution, respectively, until the algorithm first hits the optimal solution.

From Table 1, it is clear that the COMPASS algorithm with coordinate sampling is much more efficient in solving the problem than that with uniform sampling, especially when the dimension of the problem is high. When the dimension is 2, the difference between the two algorithms is negligible. When the dimension is 15, however, COMPASS with coordinate sampling evaluates fewer than 10% of the solutions and spends less than 0.1% of the computational overhead compared to COMPASS with uniform sampling. These drastic differences can be explained using the sampling efficiency shown in the last column of Table 1. When the dimension is 15, coordinate sampling finds better solutions in 58% of the iterations, while uniform sampling finds better solutions in only 4% of them.

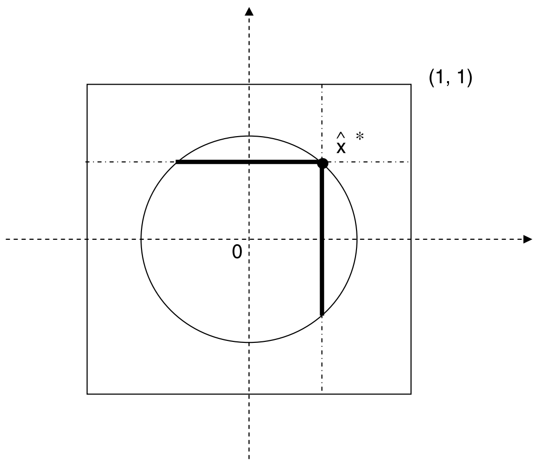

Fig. 4. The model of coordinate sampling in the MPA.

Table 1
Uniform sampling vs. coordinate sampling.

Dimension	# evaluated solutions		CPU time		% improvement	
	Uniform	Coordinate	Uniform	Coordinate	Uniform	Coordinate
2	35.9	39.8	0.0313	0.0094	88.3	83.8
5	204.4	121.2	0.489	0.048	36.1	70.6
10	1532.7	306.8	26.86	0.298	11.1	65.3
15	6375.4	503.4	1200.5	0.989	4.01	58.0
20	–	737.4	–	2.580	–	51.7
30	–	1263.9	–	22.69	–	48.1
40	–	1832.4	–	110.4	–	44.4
50	–	2409.8	–	320.5	–	41.8

With the coordinate sampling scheme, COMPASS can be applied to solve problems with much higher dimensions. As shown in Table 1, it takes on average fewer than 2500 solutions and less than 6 min to solve a 50-dimensional problem that cannot be solved using the original COMPASS algorithm in a reasonable amount of time. From this experiment, we also learn that the computational overhead of problems within 20 dimensions is generally not high when coordinate sampling is used. Therefore, the constraint pruning techniques of Xu et al. [7], which are proposed to reduce the computational overhead, may be unnecessary for problems within 20 dimensions.

3.2. Local convergence of COMPASS

When coordinate sampling is used, only solutions that differ from the sample-best solution in one coordinate can be sampled in each iteration. All solutions in the MPA that differ from the sample-best solution in at least two coordinates are no longer reachable at iteration k. When uniform sampling is used, however, all solutions in the MPA have positive probability of being sampled. This raises the question of whether the local convergence property of the original COMPASS still holds under the coordinate sampling scheme. To ensure the local convergence, Hong and Nelson [2] showed that the sampling scheme and the estimation scheme of a random search algorithm need to satisfy certain conditions. As our new algorithm only changes the sampling scheme of the original COMPASS, we show that the new sampling scheme still satisfies the required condition. The proof of the proposition is included in the Appendix.

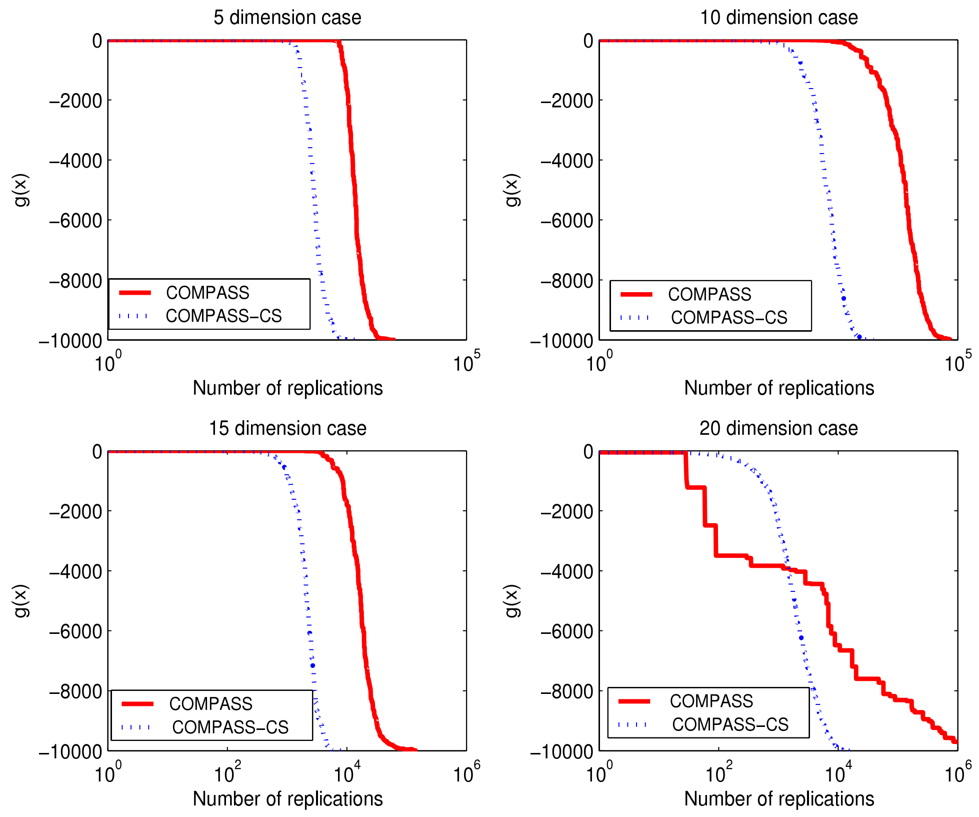

Fig. 5. The performance plot for the high-dimensional test problem: $d = 5, 10, 15$ and 20.

Proposition 1. *The coordinate sampling scheme ensures that* $\Pr\{\mathbf{x} \in \mathcal{S}_k\} > 0$ *for all* $\mathbf{x} \in \mathcal{LN}(\widehat{\mathbf{x}}^*_{k-1}) \cap \mathcal{C}_k$, *where* $\mathcal{S}_k$ *is the set of newly sampled solutions in iteration k, and* $\mathcal{LN}(\mathbf{x})$ *denotes the local neighborhood of a solution* $\mathbf{x}$ *and includes all feasible solutions that have Euclidean distance* 1 *from* $\mathbf{x}$; *i.e.,* $\mathcal{LN}(\mathbf{x}) = \{\mathbf{y} \in \Theta : \|\mathbf{x} - \mathbf{y}\| = 1\}$.

Note that the estimation scheme of the original COMPASS satisfies the required condition in [2], and Proposition 1 shows that the coordinate sampling scheme also satisfies the required condition. Then by Hong and Nelson [2] the local convergence property of the original COMPASS still holds after replacing the uniform sampling scheme by the coordinate sampling scheme.

For globally convergent random search algorithms such as the nested partitions algorithms, coordinate sampling can no longer ensure their global convergence because it is possible that all solutions along the coordinate directions from $\hat{\mathbf{x}}^*_k$ are not as good as $\hat{\mathbf{x}}^*_k$ but some other solutions are better. Therefore, the algorithms may not converge to the set of globally optimal solution even though $\hat{\mathbf{x}}^*_k$ is a locally optimal solution. To guarantee global convergence, it is important for all solutions in the MPA to be reachable in each iteration. This could be accomplished by revising the coordinate sampling scheme by first picking a random direction (not necessarily the coordinate direction) and then sampling a solution along the direction and rounding it to an integer solution (which is essentially the first step of the MIX-D sampling algorithm of Pichitlamken and Nelson [5]). Then, all solutions in the MPA are reachable and the global convergence can still be maintained.

4. Numerical results

To evaluate the performance of COMPASS with coordinate sampling, we ran the algorithm on the high-dimensional test problem in [7], which was designed to illustrate the impact of dimension. Let

$$G(\mathbf{x}) = -\beta \exp\left\{-\gamma \sum_{j=1}^{d} j(x_j - \xi^\star)^2\right\} + 0.3|g(\mathbf{x})| \cdot \epsilon,$$

where $g(\mathbf{x}) = \mathrm{E}[G(\mathbf{x})]$, $\gamma = 0.001$, $\beta = 10{,}000$, $\xi^\star = 0$, and ϵ is a standard normal random variable. The shape of the response surface of this function is like an inverted multivariate normal density function with a single globally optimal solution at $\mathbf{x} = (\xi^\star, \xi^\star, \ldots, \xi^\star)$ having value −10,000, and the stochastic noise increases near the optimal solution. Let the feasible region be the hyperbox defined by

$$x_j \in \left\{-\frac{m^{1/d}}{2}, \frac{m^{1/d}}{2}\right\}$$

for $j = 1, 2, \ldots, d$ with $m = 10^{20}$, where we round the bounds to the nearest integer if necessary. The purpose of defining the feasible region in this way is to keep the number of feasible solutions (nearly) the same as the dimension changes, isolating the impact of dimension from that of the number of feasible solutions.

We tried $d = 5, 10, 15$ and 20 and plotted the results in Fig. 5. We take the average of 25 trials for $d = 5, 10$ and 15. Because COMPASS with uniform sampling is extremely slow when $d = 20$, we only plot one sample path. We see from the plots that, although COMPASS with uniform sampling eventually finds the optimal solution in all cases, COMPASS with coordinate sampling can be significantly faster.

We have also experimented with other test problems in [7] with dimensions smaller than 6; COMPASS with coordinate sampling has at least as good performance as COMPASS with uniform random sampling.

Acknowledgements

The authors would like to thank the Area Editor Jose Blanchet and two referees for their helpful comments and suggestions. This research was partially supported by Hong Kong Research Grants Council grant CERG 613907.

Appendix

Proof of Proposition 1. Any $\mathbf{x} \in \mathcal{LN}(\widehat{\mathbf{x}}^*_{k-1}) \cap \mathcal{C}_k$ differs from $\widehat{\mathbf{x}}^*_{k-1}$ in only one of the d coordinates, i.e., along one of the d coordinate directions from $\widehat{\mathbf{x}}^*_{k-1}$. If we sample a solution using coordinate sampling at iteration k, every direction has a probability of $1/d$ being chosen and every coordinate direction has at most $|\Theta|$ feasible solutions, where $|\Theta|$ denotes the number of solutions in Θ. Thus,

$$\Pr\{\text{sampling } \mathbf{x}\} \geq \frac{1}{d|\Theta|}.$$

Because in total m solutions are sampled independently using the coordinate sampling scheme at iteration k, then for any $\mathbf{x} \in \mathcal{LN}(\widehat{\mathbf{x}}^*_{k-1}) \cap \mathcal{C}_k$,

$$\begin{aligned}\Pr\{\mathbf{x} \in \mathcal{S}_k\} &\geq 1 - (1 - \Pr\{\text{sampling } \mathbf{x}\})^m \\ &\geq 1 - \left(1 - \frac{1}{d|\Theta|}\right)^m > 0.\end{aligned}$$

This completes the proof of the proposition. □

References

[1] L.J. Hong, B.L. Nelson, Discrete optimization via simulation using COMPASS, Operations Research 54 (2006) 115–129.
[2] L.J. Hong, B.L. Nelson, A framework for locally convergent random search algorithms for discrete optimization via simulation, ACM Transactions on Modeling and Computer Simulation 19 (2007) 1–22.
[3] M.E. Kuhl, R.A. Tolentino-Peña, A dynamic crashing method for project management using simulation-based optimization, in: Proceedings of the 2008 Winter Simulation Conference, 2008, pp. 2370–2376.
[4] T.J. Petit, Supply Chain Resilience: Development of a Conceptual Framework, an Assessment Tool and an Implementation Process, Ph.D. Thesis, Ohio State University, Columbus, OH, 2008.
[5] J. Pichitlamken, B.L. Nelson, A combined procedure for optimization via simulation, ACM Transactions on Modeling and Computer Simulation 13 (2003) 155–179.
[6] L. Shi, S. Ólafsson, Nested partitions method for stochastic optimization, Methodology and Computing in Applied Probability 2 (2000) 271–291.
[7] J. Xu, B.L. Nelson, L.J Hong, Industrial strength COMPASS: a comprehensive algorithm and software for optimization via simulation, ACM Transactions on Modeling and Computer Simulation 20 (2010) 1–29.
[8] D. Yoshioka, Statistical Physics: An Introduction, Springer, Berlin, Germany, 2007.