Cooperative Optimization for Energy Minimization: A Case Study of Stereo Matching

Xiaofei Huang
School of Information Science and Technology
Tsinghua University, Beijing, P. R. China, 100084
huangxiaofei@ieee.org

Abstract

Often times, individuals working together as a team can solve hard problems beyond the capability of any individual in the team. Cooperative optimization is a newly proposed general method for attacking hard optimization problems inspired by cooperation principles in team playing. It has an established theoretical foundation and has demonstrated outstanding performances in solving real-world optimization problems. With some general settings, a cooperative optimization algorithm has a unique equilibrium and converges to it with an exponential rate regardless initial conditions and insensitive to perturbations. It also possesses a number of global optimality conditions for identifying global optima so that it can terminate its search process efficiently. This paper offers a general description of cooperative optimization, addresses a number of design issues, and presents a case study to demonstrate its power.

I. Introduction

Optimization is a core problem both in mathematics and computer science. It is a very active research area with many international conferences every year, a large amount of literature, and many researchers and users across many fields for a wide range of applications. Combinatorial optimization [1], [2] is a branch of optimization where the set of feasible solutions of problems is discrete, countable, and of a finite size. The general methods for combinatorial optimization are 1) local search [3], 2) simulated annealing [4], [5], 3) genetic algorithms [6], [7], [8], 5) ant colony optimization [9], 4) tabu search [10], 5) branch-and-bound [11], [12] 6) dynamic programming [12]. The successful applications of different combinatorial optimization methods have been reported in solving a large variety of optimization problems in practice.

Optimization is important in the areas of computer vision, pattern recognition, and image processing. For example, stereo matching is one of the most active research problems in computer vision [13], [14], [15], [16]. The goal of stereo matching is to recover the depth image of a scene from a pair of 2-D images of the same scene taken from two different locations. Like many other problems from these areas, it can be formulated as a combinatorial optimization problem, which is NP-hard [17] in computational complexity in general.

The researchers in computer vision have developed a number of search techniques which have been proven effective in practice for finding good solutions for combinatorial optimization problems. Two well-known ones are the cooperative algorithm proposed by D. Marr and T. Poggio in [16] for stereo matching and the probabilistic relaxation proposed by A. Rosenfield et al [18] for scene labeling.

Recently, there are some remarkable progresses in discovering new optimization methods for solving computer vision problems. Graph cuts [14], [19], [13], [20] is a powerful specialized optimization technique popular in computer vision. It has the best known results in energy minimization in the two recent evaluations of stereo algorithms [13], [21], more powerful than the classic simulated annealing method. However, graph cuts has a limitation in its scope because it is only applicable when the energy minimization of a vision problem can be reduced into a problem of finding the minimum cut in a graph [20].

The second optimization method is so called the sum-product algorithm [22], a generalized belief propagation algorithm developed in AI [23]. The sum-product algorithm is the most powerful optimization method ever found so far for attacking hard optimization problems raised from channel decoding in communications. The min-sum algorithm and max-product algorithm [24], [25] are its variations. It has also been successful applied to solve several computer vision problems with promising experimental results [26].

The third method proposed recently is so called max-product tree-reweighted message passing [27]. It is based on a lower bounding technique called linear programming relaxation. Its improvement has been proposed recently and its successful applications in computer vision have been reported [28].

The cooperative optimization is a newly discovered general optimization method for attacking hard optimization problems [29], [30], [31], [32]. It has been found in the experiments [33], [34], [35], [36], [37] that cooperative optimization has achieved remarkable performances at solving a number of real-world NP-hard problems with the number of variables ranging from thousands to hundreds of thousands. The problems span several areas, proving its generality and power.

For example, cooperative optimization algorithms have been proposed for DNA image analysis [33], shape from shading [32], stereo matching [30], [34], and image segmentation [38]. In the second case, it significantly outperformed the classic simulated annealing in finding global optimal solutions. In the third case, its performance is comparable with graph cuts in terms of solution quality, and is twice as faster as graph cuts in software simulation using the common evaluation framework for stereo matching [13]. In the fourth case, it is ten times faster than graph cuts and has reduced the error rate by two to three factors. In all these cases, its memory usage is efficient and fixed, its operations are simple, regular, and fully scalable. All these features make it suitable for parallel hardware implementations.

This paper is organized in three major themes as 1) a formal presentation for cooperative optimization, 2) design issues, and 3) a case study. They are the generalization and the extension of the previous papers on cooperative optimization. In the case study, another cooperative optimization algorithm for stereo matching besides the one proposed before [30], [34] is offered to demonstrate the power and flexibility of cooperative optimization. Compared with the previous one for stereo matching, the new one lowers the energy levels of solutions further and is more than ten times faster. Just like the previous one, the new one is also simple in computation and fully parallel in operations, suitable for hardware implementations.

II. Cooperative Multi-Agent System for Distributed Optimization

Different forms of cooperative optimization can be derived from different cooperation schemes. The basic form defines an important collection of cooperative optimization algorithms. There are two different ways to derive it; namely, 1) as a cooperative multi-agent system for distributed optimization and 2) as a lower bounding technique for finding global optimums. Each way offers its own inspirations and insights to understand the algorithms. This section describes the first way. The following section offers the description for the second way. Readers who are not interested in them can directly jump to Section V for a general description of cooperative optimization. Those three sections are relatively independent to each other.

A. Inspiration and Basic Ideas

Team playing is a common social behavior among individuals of the same species (or different) where the team members working together can achieve goals or solve hard problems which are beyond the capability of any member in the team. Often times, team playing is achieved through competition and cooperation among the members in a team. Usually, competition or cooperation alone can hardly lead to good solutions either for a team or for the individuals in the team. Without competition, individuals in a team may lose motivation to pursue better solutions. Without cooperation, they might directly conflict with each other and poor solutions might be reached both for the team and themselves. Through properly balanced competition and cooperation, individuals in a team can find the best solutions for the team and possibly good solutions for themselves at the same time.

In the terms of computer science, we can view a team of this kind as a cooperative system with multiple agents. In the system, each agent has its own objective. The collection of all the agent's objectives form the objective of the system. We can use a cooperative system to solve a hard optimization problem following the divide-and-conquer principle. We first break up the objective function of the optimization problem into a number of sub-objective functions of manageable sizes and complexities. Following that, we assign each sub-objective function to an agent in a system as the agent's own objective function and ask those agents in the system to optimize their own objective functions through competition and cooperation. (Throughout this paper, we use the term "objective" and "objective function" interchangeably since the objective of an optimization problem is defined by an objective function and this paper focuses only on optimizing objective functions.)

Specifically, the competition is achieved by asking each agent to optimize its own objective function by applying problem-specific optimization methods or heuristics. However, the objectives of agents may not be always aligned with each other. In other words, the best solutions of the agents for optimizing their own objective functions may conflict with each other. To resolve the conflicts, each agent passes its solution to its neighbors through local message passing. After receiving its neighbor's solutions, each agent compromises its solution with the solutions of its neighbors. The solution compromising is achieved by modifying the objective

```
Initialization
    For each individual $i$ in the system, find the initial solution,
        $Find\_Solution(Objective(i)) \Rightarrow Solution(i, t = 0);$
Iteration
    For each individual $i$ in the system,
        Modify its original objective by including its neighbors' solutions,
            $Modify\_Objective(Objective(i), \{Solution(j, t)|j \in Neighbors(i)\})$
                $\Rightarrow Objective(i, t + 1);$
        Find solutions of the modified objective,
            $Find\_Solution(Objective(i, t + 1)) \Rightarrow Solution(i, t + 1);$
```

Fig. 1. Cooperative Multi-Agent System for Distributed Optimization.

function of each agent to take into account its neighbors' solutions. It is important to note that solution compromising among agents is a key concept for understanding the cooperation strategy introduced by cooperative optimization.

Let the objective of the individual i be $Objective(i)$. Let the solution of the individual i at time t be $Solution(i, t)$. Let the collection of solutions of the neighbors of the individual i at time t be $\{Solution(j, t)|j \in Neighbors(i)\}$. The basic operations of a cooperative system are organized as a process shown in Figure 1.

The process of a cooperative system of this kind is iterative and self-organized and each agent in the system is autonomous. The system is also inherently distributed and parallel, making the entire system highly scalable and less vulnerable to perturbations and disruptions on individuals than a centralized system. Despite of its simplicity, it has many interesting emerging behaviors and can attack many challenging optimization problems.

B. Basic Form of Cooperative Optimization

In light of the cooperative multi-agent system for distributed optimization described in Fig. 1, we can derive the basic form of cooperative optimization now. It is based on a direct way for defining the solution of each agent and a simple way to modify the objective of each agent.

The derivation can be generalized further in a straightforward way to any other definitions of solutions and modifications of objectives.

Given a multivariate objective function $E(x_1, x_2, \ldots, x_n)$ of n variables, or simply denoted as $E(x)$, where each variable x_i is of a finite domain D_i of size $|D_i|$. Assume that $E(x)$ can be decomposed into n sub-objective functions $E_i(x)$, denoted as $\{E_i(x)\}$, satisfying

1) $E(x) = E_1(x) + E_2(x) + \ldots + E_n(x)$,
2) $E_i(x)$, for $i = 1, 2, \ldots, n$, contains at least variable x_i,
3) the minimization of $E_i(x)$, for $i = 1, 2, \ldots, n$, is computationally manageable in complexity.

Let us assign $E_i(x)$ as the objective of agent i,

$$Objective(i) = E_i(x), \quad \text{for } i = 1, 2, \ldots, n \ .$$

There are n agents in the system, one agent for each sub-objective function.

Let the initial solution of agent i be the minimization result of $E_i(x)$ defined as follows,

$$Solution(i, t = 0) = \min_{X_i \backslash x_i} E_i(x) \ ,$$

where X_i is the set of variables contained in $E_i(x)$, and $\min_{X_i \backslash x_i}$ stands for minimizing with respect to all variables in X_i excluding x_i. The solution is an unary function on variable x_i, denoted as $\Psi_i^{(0)}(x_i)$.

Assume that the system takes discrete-time with iteration step $k = 1, 2, 3, \ldots$. To simplify notations, let $\tilde{E}_i^{(k)}(x)$ be the modified objective function of agent i at iteration k, i.e.,

$$\tilde{E}_i^{(k)}(x) = Objective(i, t = k) \ .$$

It is also referred to as the i-th modified sub-objective of the system. The agent's solution at the iteration is defined as

$$Solution(i, t = k) = \min_{X_i \backslash x_i} \tilde{E}_i^{(k)}(x) \ . \tag{1}$$

The solution is an unary function on variable x_i, denoted as $\Psi_i^{(k)}(x_i)$. It is the state of agent i at iteration k. It can be represented as a vector of real values of size $|D_i|$, the domain size of variable x_i. The i-th equation in (1) defines the dynamics of agent i. All the n equations define the dynamics of the system.

As described in the previous subsection, the cooperation among the agents in the system is introduced by solution compromising via modifying the objective of each agent. Let agent i define its modified objective function $\tilde{E}_i^{(k)}(x)$ at iteration k as a linear combination of its original objective $E_i(x)$ and the solutions of its neighbors at the previous iteration $k-1$ as follows,

$$\tilde{E}_i^{(k)}(x) = (1-\lambda_k)\, E_i(x) + \lambda_k \sum_{j \in Neighbors(i)} w_{ij} \Psi_j^{(k-1)}(x_j) \ , \tag{2}$$

where λ_k and w_{ij} are coefficients of the linear combination.

Agent j is the neighbor of agent i if variable x_j of the same index j is contained in the agent i's objective function $E_i(x)$. (Based on this definition, the agent i is also a neighbor of itself. Such a generalization is necessary because there is no restriction to have agent i modify its objective using its own solution.) The neighbors of agent i is denoted as $\mathcal{N}(i)$, i.e., $\mathcal{N}(i) = Neighbors(i)$. Specifically, it is defined as the set of indices as

$$\mathcal{N}(i) = \{j | \{x_j\} \in X_i\} \ .$$

Substituting Eq. (2) into Eq. (1) and letting $w_{ij} = 0$ if $j \notin \mathcal{N}(i)$, the dynamics of the cooperative system can be written as the following n difference equations,

$$\Psi_i^{(k)}(x_i) = \min_{X_i \setminus x_i} \left((1-\lambda_k)\, E_i(x) + \lambda_k \sum_j w_{ij} \Psi_j^{(k-1)}(x_j) \right), \quad \text{for } i = 1, 2, \ldots, n \ . \tag{3}$$

Such a set of difference equations defines a basic cooperative optimization system (algorithm) for minimizing an objective function of the form $\sum_i E_i(x)$.

At iteration k, variable x_i, for $i = 1, 2, \ldots, n$, has a value in the solution for minimizing the i-th modified sub-objective function $\tilde{E}_i^{(k)}(x)$. It is denoted as $\tilde{x}_i^{(k)}$, i.e.,

$$\tilde{x}_i^{(k)} = \arg\min_{x_i} \min_{X_i \setminus x_i} \tilde{E}_i^{(k)}(x) \ .$$

From (1), we have

$$\tilde{x}_i^{(k)} = \arg\min_{x_i} \Psi_i^{(k)}(x_i) \ . \tag{4}$$

The agent i is responsible for assigning that value to variable x_i. The assignments of other variables are taken care of by other agents. All these values together form a solution of the system at iteration k, denoted as $\tilde{x}^{(k)}$.

Putting everything together, we have the pseudo code of the algorithm is given in Figure 2. The global optimality condition mentioned in the line 7 will be discussed in detail later in this paper.

```
**Procedure** Basic Cooperative Optimization Algorithm
1  Initialize the soft assignment function $\Psi_i^{(0)}(x_i)$, for each $i$;
2  for $k := 1$ to $max\_iteration$ do
3      for each $i$ do
           /* modify the $i$-th sub-objective function $E_i(x)$ */
4              $\tilde{E}_i^{(k)}(x) := (1-\lambda_k)E_i(x) + \lambda_k \sum_j w_{ij}\Psi_i^{(k-1)}(x_i)$ ;
           /* minimize the modified sub-objective function */
5              $\Psi_i^{(k)}(x_i) := \min_{X_i \setminus x_i} \tilde{E}_i^{(k)}(x)$;
           /* find the best value for $x_i$ */
6              $\tilde{x}_i^{(k)} := \arg\min_{x_i} \Psi_i^{(k)}(x_i)$;
7      if $\tilde{x}^{(k)}$ is a global optimal solution return $\tilde{x}^{(k)}$;
8  return $\tilde{x}^{(k)}$; /* as an approximate solution */
```

Fig. 2. Basic cooperative optimization algorithm for minimizing an objective function of the form $E(x) = \sum_{i=1}^{n} E_i(x)$.

C. Cooperation Strength and Propagation Matrix

The coefficient λ_k in (3) controls the level of the cooperation among the agents at iteration k. It is so called the cooperation strength, satisfying $0 \leq \lambda_k < 1$. From (3) we can see that, for each agent, a high value for λ_k will weigh the solutions of the other agents more than its own objective $E_i(x)$. In other words, the agents in the system tend to compromise more with their solutions. As a consequence, a strong level of cooperation is reached in this case. If the cooperation strength λ_k is of a small value, the cooperation among the agents is weak. Particularly, if it is equal to zero, there is no cooperation among the agents and each agent minimizes its own objective function independently (see (3)).

The coefficients w_{ij} control the propagation of solutions $\Psi_j^{(k-1)}(x_j)$, for $j = 1, 2, \ldots, n$, as messages among the agents in the system. All w_{ij}s together form a $n \times n$ matrix called the propagation matrix. To have $\sum_i E_i(x)$ as the objective function to be minimized, it is required [33] that the propagation matrix $W = (w_{ij})_{n \times n}$ is non-negative and

$$\sum_{i=1}^{n} w_{ij} = 1, \quad \text{for } j = 1, 2, \ldots, n \ .$$

To have solutions $\Psi_j^{(k-1)}(x_j)$ uniformly propagated among all the agents, it is required [33] that the propagation matrix W is irreducible. A matrix W is called reducible if there exists a permutation matrix P such that PWP^T has the block form

$$\begin{pmatrix} A & B \\ O & C \end{pmatrix} .$$

The role of propagation matrices in basic cooperative optimization algorithms is exactly same as the one of transition matrices in Markov chains (or random walks over directed graphs). In a Markov chain, a transition matrix governs the distribution of states over time. In a basic cooperative optimization algorithm, a propagation matrix governs the distribution of solutions among agents. The mathematical foundation for analyzing Markov chains has been well established. They can be directly applied to analyze the message propagation of cooperative optimization.

D. Soft Decisions as Messages Passed Among Agents

As mentioned before, the solution $\Psi_i^{(k)}(x_i)$ of agent i at iteration k is an unary function on x_i storing the solution of minimizing the i-th modified sub-objective function $\tilde{E}_i^{(k)}(x)$ (see (1)). Given a value of x_i, $\Psi_i^{(k)}(x_i)$ is the minimal value of $\tilde{E}_i^{(k)}(x)$ with the variable x_i fixed to that value. To minimize $\tilde{E}_i^{(k)}(x)$, the values of x_i which have smaller function values $\Psi_i^{(k)}(x_i)$ are preferred more than those of higher function values. The best value for assigning the variable x_i is the one of the minimal function value $\Psi_i^{(k)}(x_i)$ (see (4)). Therefore, $\Psi_i^{(k)}(x_i)$ is inversely related to the preferences over different values of x_i for minimizing $\tilde{E}_i^{(k)}(x)$. It is so called the assignment constraint on variable x_i, an algorithm introduced constraint on the variable. It can also be viewed as a soft decision made by the agent for assigning the variable x_i at iteration k.

In particular, a soft decision of agent i falls back to a hard decision for assigning the variable x_i when the agent accept only one value and reject all the rest values. Such a hard decision can be represented by the assignment constraint $\Psi_i^{(k)}(x_i)$ as $\Psi_i^{(k)}(\tilde{x}_i) = 0$, for some $\tilde{x}_i \in D_i$, and $\Psi_i^{(k)}(x_i) = \infty$ for any $x_i \neq \tilde{x}_i$.

With that insight, it can be understood now that the messages propagated around among the agents in a basic cooperative optimization system are the soft decisions for assigning variables. An agent can make a better decision using soft decisions propagated from its neighbors than using the hard ones instead. It is important to note that soft decision making is a critical feature of cooperative optimization, which makes it fundamentally different from many classic optimization methods where hard decisions are made for assigning variables.

E. A Simple Example

Given an objective function of the following form

$$\begin{aligned} &E(x_1, x_2, x_3, x_4, x_5) = \\ &\quad f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4) + f_5(x_5) + \\ &\quad f_{1,2}(x_1, x_2) + f_{2,3}(x_2, x_3) + f_{3,4}(x_3, x_4) + \\ &\quad f_{4,5}(x_4, x_5) + f_{1,5}(x_1, x_5) + f_{2,5}(x_2, x_5) \ , \end{aligned} \tag{5}$$

where each variable is of a finite domain. The goal is to seek values (labels) of the five variables such that the objective function is minimized.

Let us simply denote the function as

$$\begin{aligned} E(x) = f_1 + f_2 + f_3 + f_4 + f_5 + \\ f_{1,2} + f_{2,3} + f_{3,4} + f_{4,5} + f_{1,5} + f_{2,5} \ . \end{aligned}$$

To design a basic cooperative optimization algorithm to minimize the objective function, we first decompose it into the following five sub-objective functions,

$$\begin{aligned} E_1(x_1, x_2, x_5) &= f_1 + f_{1,2}/2 + f_{1,5}/2; \\ E_2(x_1, x_2, x_3, x_5) &= f_2 + f_{1,2}/2 + f_{2,3}/2 + f_{2,5}/2; \\ E_3(x_2, x_3, x_4) &= f_3 + f_{2,3}/2 + f_{3,4}/2; \\ E_4(x_3, x_4, x_5) &= f_4 + f_{3,4}/2 + f_{4,5}/2; \\ E_5(x_1, x_2, x_4, x_5) &= f_5 + f_{1,5}/2 + f_{2,5}/2 + f_{4,5}/2. \end{aligned}$$

A propagation matrix W of dimensions 5×5 can be chosen as

$$W = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & 0 & \frac{1}{2} & 0 \end{pmatrix} \tag{6}$$

With the decomposition and the propagation matrix, substituting them into (3) we have a basic cooperative optimization algorithm with five difference equations for minimizing the five sub-objective functions in an iterative and cooperative way.

F. Basic Canonical Form as Generalization

Replacing $\Psi_i^{(k)}(x)$ by $(1-\lambda_k)\Psi_i^{(k)}(x)$ in the difference equations (3), we have the basic canonical form of cooperative optimization as

$$\Psi_i^{(k)}(x_i) = \min_{X_i \setminus x_i} \left(E_i(x) + \lambda_k \sum_j w_{ij} \Psi_j^{(k-1)}(x_j) \right) , \quad \text{for } i = 1, 2, \ldots, n \ . \tag{7}$$

The basic form of cooperative optimization (3) has its cooperation strength λ_k restricted to $0 \le \lambda_k < 1$. It is because its difference equations (3) do not make sense when $\lambda_k \ge 1$. However, such a restriction can be relaxed to $0 \le \lambda_k$ for the basic canonical form (7). Often in times in practice, the basic canonical form is preferred over the basic one because the cooperation strength λ_k in the former has a broader range to choose from to maximize performance.

III. Cooperative Optimization as Lower Bounding Technique

A. Bound Function Tightening Technique for Optimization

In principle, a basic cooperative optimization algorithm can be understood as a lower bounding technique for finding global minima. It first initializes a function of some form as a lower bound function to an objective function. One may intentionally choose a form for the lower bound function such that the minimization of the function is simple in computation. Following that, the algorithm progressively tightens the lower bound function until its global minimum touches the global minimum of the original objective function. The latter is then found by searching the former instead (see the illustration in Fig. 3).

Specifically, let the objective function to be minimized be $E(x)$. Assume that the initial lower bound function be $E_-^{(0)}(x)$, $E_-^{(0)}(x) \le E(x)$. From $E_-^{(0)}(x)$, assume that the algorithm progressively tightens the function in an iterative way such that

$$E_-^{(0)}(x) \le E_-^{(1)}(x) \le \ldots \le E_-^{(k)}(x) \le E(x) \ ,$$

where k is the iteration number.

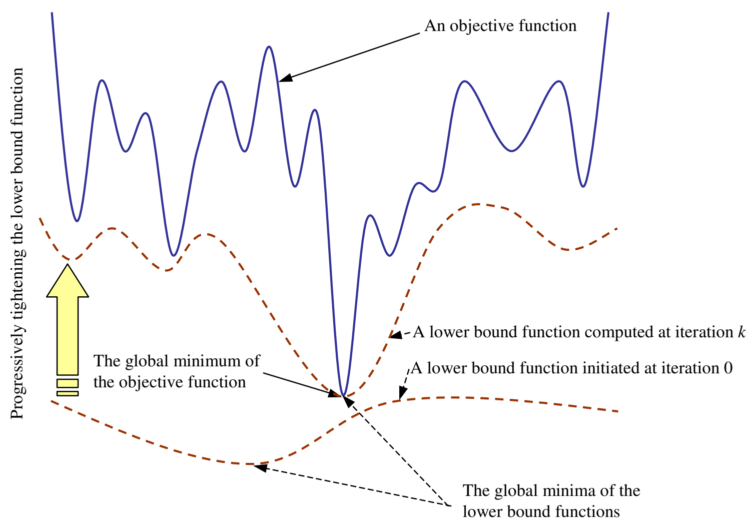

An objective function
Progressively tightening the lower bound function
A lower bound function computed at iteration k
A lower bound function initiated at iteration 0
The global minimum of the objective function
The global minima of the lower bound functions

Fig. 3. The global minimum of a complex multivariate objective function can be found by progressively tightening a lower bound function of some simple form until its global minimum touches the one of the original objective function.

Let the global minimum of the lower bound function $E_{-}^{(k)}(x)$ at iteration k be $\tilde{x}^{(k)}$. Finding $\tilde{x}^{(k)}$ is simple in computation due to the simple form of the lower bound function $E_{-}^{(k)}(x)$. At iteration k, if the algorithm found that the lower bound function $E_{-}^{(k)}(x)$ at the solution $\tilde{x}^{(k)}$ has the same function value as the original objective function $E(x)$, i.e.,

$$E_{-}^{(k)}(\tilde{x}^{(k)}) = E(\tilde{x}^{(k)}) \ .$$

In other words, the two functions touch each other at the point where $x = \tilde{x}^{(k)}$ in the search space. Then $\tilde{x}^{(k)}$ must also be the global minimum of $E(x)$ simply because

$$E(\tilde{x}^{(k)}) = E_{-}^{(k)}(\tilde{x}^{(k)}) \le E_{-}^{(k)}(x) \le E(x), \quad \text{for any } x \ . \tag{8}$$

Such a condition implies that the lower bound function $E_{-}^{(k)}(x)$ has been tightened enough such that its global minimum $\tilde{x}^{(k)}$ touches the global minimum of the original objective function $E(x)$. The latter is thus found by searching the former instead.

Such a lower bounding technique is so called the bound function tightening technique for optimization. There are other lower bounding techniques based on principles different from this one. Examples are Lagrangian relaxation techniques, cutting plane techniques, branch-and-bound algorithms, and branch-and-cut algorithms.

B. Basic Form as Lower Bounding Technique

In light of the bound function tightening technique described in the previous subsection, we can derive the basic form of cooperative optimization based on a simple form of lower bound functions. The derivation can be generalized further in a straightforward way to any other forms of lower bound functions.

Given an objective function of n variables, $E(x_1, x_2, \ldots, x_n)$, or simply denoted as $E(x)$. Assume that $E_-(x)$ is a lower bound function of $E(x)$ defined on the same set of variables. Obviously the linear combination of the two functions,

$$(1-\lambda)E(x) + \lambda E_-(x) \ , \tag{9}$$

defines a new lower bound function of $E(x)$ if the parameter λ satisfying $0 \leq \lambda < 1$.

Let us choose a simple form for the lower bound function as

$$E_-(x) = \Psi_1(x_1) + \Psi_2(x_2) + \ldots + \Psi_n(x_n) \ , \tag{10}$$

where $\Psi_i(x_i)$ is an unary component function defined on variable x_i, for $i = 1, 2, \ldots, n$. Its global minimum, denoted as $\tilde{x}$, can be easily found by minimizing the unary component functions $\Psi_i(x_i)$ independently as

$$\tilde{x}_i = \arg\min_{x_i} \Psi_i(x_i), \quad \text{for } i = 1, 2, \ldots, n \ .$$

Assume that the objective function $E(x)$ can be decomposed into n sub-objective functions,

$$E(x) = E_1(x) + E_2(x) + \ldots + E_n(x) \ .$$

The lower bound function $E_-(x)$ can also be easily decomposed into n sub-functions as follows

$$E_-(x) = \sum_{i=1}^{n} w_{ij}\Psi_j(x_j), \quad \text{where}$$

$$w_{ij} \geq 0 \text{ and } \sum_i w_{ij} = 1, \quad \text{for } 1 \leq i, j \leq n \ .$$

Based on the two decompositions, the new lower bound function (9) can be rewritten as

$$\sum_{i=1}^{n} \left((1-\lambda)E_i(x) + \lambda \sum_j w_{ij}\Psi_j(x_j) \right) \ . \tag{11}$$

To put the above function in a simple form, let

$$\tilde{E}_i(x) = (1-\lambda)E_i(x) + \lambda \sum_j w_{ij}\Psi_j(x_j) \ .$$

Then it can be rewritten simply as

$$\sum_{i=1}^{n} \tilde{E}_i(x) \ .$$

In the above sum, let $\tilde{X}_i$ be the set of variables contained in the i-th component function $\tilde{E}_i(x)$. If we minimize the function with respect to all variables in $\tilde{X}_i$ except for x_i, we obtain an unary function defined on x_i, denoted as $\Psi'_i(x_i)$, i.e.,

$$\Psi'_i(x_i) = \min_{\tilde{X}_i \backslash x_i} \tilde{E}_i(x) \ . \tag{12}$$

The sum of those unary functions defines another lower bound function of $E(x)$, denoted as $E'_-(x)$, i.e.,

$$E'_-(x) = \sum_{i=1}^{n} \Psi'_i(x_i) \leq E(x) \ .$$

This new lower bound function has exactly the same form as the original one $E_-(x) = \sum_i \Psi_i(x_i)$. Therefore, from a lower bound function $E_-(x)$ of the form $\sum_i \Psi_i(x_i)$, we can compute another lower bound function $E'_-(x)$ of the same form. Such a process can be repeated and we can have an iterative algorithm to compute new lower bound functions.

Rewriting Eq. (12) in an iterative format, we have

$$\Psi_i^{(k)}(x_i) = \min_{\tilde{X}_i \backslash x_i} \left((1 - \lambda_k)\, E_i(x) + \lambda_k \sum_j w_{ij} \Psi_j^{(k-1)}(x_j) \right) , \tag{13}$$

where k is the iteration step, $k = 1, 2, 3, \ldots$. The above n difference equations define a basic cooperative optimization algorithm for minimizing an objective function $E(x)$ of the form $\sum_i E_i(x)$.

The solution at iteration k, denoted as $\tilde{x}^{(k)}$, is defined as the global minimal solution of the lower bound function $E_-^{(k)}(x)$ at the iteration, i.e.,

$$\tilde{x}^{(k)} = \arg\min_x E_-^{(k)}(x),$$

which can be easily obtained as

$$\tilde{x}_i^{(k)} = \arg\min_{x_i} \Psi_i^{(k)}(x_i), \quad \text{for } i = 1, 2, \ldots, n \ . \tag{14}$$

If $E_-^{(k)}(\tilde{x}^{(k)}) = E(\tilde{x}^{(k)})$ at some iteration k, then the solution $\tilde{x}^{(k)}$ must be the global minimum of the original objective function $E(x)$.

Without loss of generality, we assume in the following discussions that all sub-objective functions $E_i(x)$ are nonnegative ones. One may choose the initial condition as $\Psi_i^{(0)}(x_i) = 0$, for any value of x_i and $i = 1, 2, \ldots, n$. The parameter λ_k can be varied from one iteration to another iteration. If it is of a constant value and the above initial condition has been chosen, cooperative optimization theory [33] tells us that the lower bound function $E_-^{(k)}(x)$ is monotonically non-decreasing as shown in (8).

IV. Computational Properties

A. General Convergence Properties of Cooperative Optimization

It has been shown that a basic cooperative optimization algorithm (3) has some important computational properties [33]. Given a constant cooperation strength λ, i.e., $\lambda_k = \lambda$ for all ks, the algorithm has one and only one equilibrium. It always converges to the unique equilibrium with an exponential rate regardless of initial conditions and perturbations. The two convergence theorems proved in [33] are very important and so they are listed here again. One formally describes the existence and the uniqueness of the equilibrium of the algorithm, and the another reveals the convergence property of the algorithm.

Theorem 4.1: A basic cooperative optimization algorithm with a constant cooperation strength λ ($0 \leq \lambda < 1$) has one and only one equilibrium. That is, the difference equations (3) of the algorithm have one and only one solution (equilibrium), denoted as a vector $(\Psi_1^{(\infty)}, \Psi_2^{(\infty)}, \ldots, \Psi_n^{(\infty)})^T$, or simply $\Psi^{(\infty)}$.

Theorem 4.2: A basic cooperative optimization algorithm with a constant cooperation strength λ ($0 \leq \lambda < 1$) converges exponentially to its unique equilibrium $\Psi^{(\infty)}$ with the rate λ with any choice of the initial condition $\Psi^{(0)}$. That is,

$$\|\Psi^{(k)} - \Psi^{(\infty)}\|_\infty \leq \lambda^k \|\Psi^{(0)} - \Psi^{(\infty)}\|_\infty \ . \tag{15}$$

where $\|x\|_\infty$ is the maximum norm of the vector x defined as

$$\|x\|_\infty = \max_i |x_i| \ .$$

The two theorems indicate that every basic cooperative optimization algorithm (3) is stable and has a unique attractor, $\Psi^{(\infty)}$. Hence, the evolution of the algorithms is robust, insensitive to perturbations. The final solution of the algorithms is independent of their initial conditions. In contrast, the conventional algorithms based on iterative local improvement of solutions may have many local attractors due to the local minima problem. The evolution of those local optimization algorithms are sensitive to perturbations, and the final solution of those algorithms is dependent on their initial conditions.

Furthermore, the basic cooperative optimization algorithms (3) possess a number of global optimality conditions for identifying global optima. They know whether a solution they found is a global optimum so that they can terminate their search process efficiently. However, this statement does not imply that *NP=P* because a basic cooperative optimization algorithm can only verify within a polynomial time whether a solution it found is a global optimum or not. It cannot decide the global optimality for any given solution other than those it found.

It is important to note that a basic canonical cooperative optimization algorithm (7) may no longer possess the unique equilibrium property when its cooperation strengths at some iterations are greater than one, i.e., $\lambda_k > 1$ for some ks. In this case, the algorithm may have multiple equilibriums. It can evolve into any one of them depending on its initial settings of the assignment constraints $\Psi_i^{(0)}(x_i)$ $(1 \leq i \leq n)$.

B. Consensus Solution and Solution Consensus in Distributed Optimization

As described before, a basic cooperative optimization algorithm is defined by the n difference equations (3). The i-th equation defines the minimization of the i-th modified sub-objective function $\tilde{E}_i^{(k)}(x)$ (defined in (2)). Given any variable, say x_i, it may be contained in several modified sub-objective functions. At each iteration, x_i has a value in the optimal solution for minimizing each of the modified sub-objective functions containing the variable. Those values may not be the same. If all of them are of the same value at some iteration, we say that the cooperative optimization algorithm reach a consensus assignment for that variable. Moreover, if a consensus assignment is reached for every variable of the problem at hand at some iteration, we call the minimization of the n modified sub-objective functions reaches a solution consensus. That is, there is no conflict among the solutions in terms of variable assignments for minimizing those functions. In this case, those consensus assignments form a solution, called a consensus solution, and the algorithm is called reaching a consensus solution.

To be more specific, given n modified sub-objective functions, $\tilde{E}_i(x)$, for $i = 1, 2, \ldots, n$ (to simplify notation, let us drop the superscript k temporarily). Let the optimal solution of the i-th modified sub-objective function be $\tilde{x}(\tilde{E}_i)$, i.e.,

$$\tilde{x}(\tilde{E}_i) = \arg\min_x \tilde{E}_i(x) \ .$$

Assume that variable x_i is contained in both j-th and k-th modified sub-objective functions $\tilde{E}_j(x)$, $\tilde{E}_k(x)$. However, it is not necessary that

$$\tilde{x}_i(\tilde{E}_j) = \tilde{x}_i(\tilde{E}_k) \ .$$

Given a variable x_i, if the above equality holds for any j and k where $\tilde{E}_j(x)$ and $\tilde{E}_k(x)$ contain x_i, then a consensus assignment is reached for that variable with the assignment value denoted as $\tilde{x}_i$. Moreover, if the above statement is true for any variable, we call the minimization for all $\tilde{E}_i(x)$s reaches a solution consensus. The solution $\tilde{x}$ with $\tilde{x}_i$ as the value of variable x_i is called a consensus solution.

As defined before, $\tilde{X}_i$ stands for the set of variables contained in the function $\tilde{E}_i(x)$. $\tilde{X}_i$ is a subset of variables, i.e., $\tilde{X}_i \subseteq X = \{x_1, x_2, \ldots, x_n\}$. Let $\tilde{x}(\tilde{X}_i)$ stand for the restriction of a solution $\tilde{x}$ on $\tilde{X}_i$. Another way to recognize a consensus solution $\tilde{x}$ is to check if $\tilde{x}(\tilde{X}_i)$, for any i, is the global minimum of $\tilde{E}_i(x)$, i.e.,

$$\tilde{x}(\tilde{X}_i) = \arg\min_x \tilde{E}_i(x), \quad \text{for any } i \ .$$

Simply put, a solution is a consensus one if it is the global minimum of every modified sub-objective function.

C. Consensus Solution in Cooperative Optimization

Consensus solution is an important concept of cooperative optimization. If a consensus solution is found at some iteration or iterations, then we can find out the closeness between the consensus solution and the global optimal solution in cost. The following theorem from [33] makes these points clearer.

Theorem 4.3: Let

$$E_{-}^{*(k)} = \sum_{i=1}^{n} \Psi_i^{(k)}(\tilde{x}_i^{(k)}), \quad \text{where } \tilde{x}_i^{(k)} = \arg\min_{x_i} \Psi_i^{(k)}(x_i) \ .$$

Given any propagation matrix W, and the general initial condition $\Psi_i^{(0)}(x_i) = 0$, for each i, or $\lambda_1 = 0$. If a consensus solution $\tilde{x}$ is found at iteration k_1 and remains the same from iteration

k_1 to iteration k_2, then the closeness between the cost of $\tilde{x}$, $E(\tilde{x})$, and the optimal cost, E^*, satisfies the following two inequalities,

$$0 \leq E(\tilde{x}) - E^* \leq \left(\prod_{k=k_1}^{k_2} \lambda_k \right) \left(E(\tilde{x}) - E_-^{*(k_1-1)} \right) , \tag{16}$$

$$0 \leq E(\tilde{x}) - E^* \leq \frac{\prod_{k=k_1}^{k_2} \lambda_k}{1 - \prod_{k=k_1}^{k_2} \lambda_k} (E^* - E_-^{*(k_1-1)}) \ , \tag{17}$$

where $(E^* - E_-^{*(k_1-1)})$ is the difference between the optimal cost E^* and the lower bound on the optimal cost $E_-^{*(k_1-1)}$ obtained at iteration $k_1 - 1$.

In particular, if $1 - \lambda_k \geq \epsilon > 0$, for $k_1 \leq k \leq k_2$, when $k_2 - k_1 \to \infty$,

$$E(\tilde{x}) \to E^* \ .$$

That is, the consensus solution $\tilde{x}$ must be global minimum of $E(x)$, i.e.,$\tilde{x} = x^*$.

Consensus solution is also an important concept of cooperative optimization for defining global optimality conditions. The cooperative optimization theory tells us that a consensus solution can be the global minimal solution. As mentioned in the previous subsection that a basic cooperative optimization algorithm has one and only one equilibrium given a constant cooperation strength. If a cooperative optimization algorithm reaches an equilibrium after some number of iterations and a consensus solution is found at the same time, then the consensus solution must be the global minimal solution, guaranteed by theory. The following theorem (with its proof in the appendix) establishes the connection between a consensus solution and a global optimal solution.

Theorem 4.4: Assume that a basic cooperative optimization (3) reaches its equilibrium at some iteration, denoted as $\Psi^{(\infty)}$. That is, $\Psi^{(\infty)}$ is a solution to the difference equations (3). If a consensus solution $\tilde{x}$ is found at the same iteration, then it must be the global minimum of $E(x)$, i.e.,$\tilde{x} = x^*$.

Besides the basic global optimality condition given in the above theorem, a few more ones are offered in [33] for identifying global optimal solutions. The capability of recognizing global optimums is a critical property for any optimization algorithm. Without any global optimality condition, it will be hard for an optimization algorithm to know where to find global optimal solutions and whether a solution it found is a global optimum. Finding ways of identifying global optimums for any optimization algorithm is of both practical interests as well as theoretical importance.

D. Further Generalization of Convergence Properties

The convergence theorem 4.3 can be generalized further to any initial conditions for $\Psi^{(0)}$ and λ_1, and to any cooperation strength series $\{\lambda_k\}_{k\geq 1}$. Dropping the restriction on the initial conditions $\Psi^{(0)}$ and λ_1 in the theorem, from the difference equations (3), we have

$$E^* - E_-^{*(k_2)} = \left(\prod_{k=k_1}^{k_2} \lambda_k\right) \left(E^* - E_-^{*(k_1-1)}\right) \ . \tag{18}$$

It is obvious from the above equation that $E_-^{*(k_2)}$ still approaches E^* exponentially with the rate λ when the cooperation strength λ_k is of a constant value λ ($0 \leq \lambda < 1$).

When the cooperation strength λ_k is not of a constant value λ, the convergence to the global optimum is still guaranteed as long as the cooperation strength series $\{1-\lambda_k\}_{k\geq 1}$ is divergent.

Lemma 4.1 (Infinite Products): Let $\{\lambda_k\}_{k\geq 1}$ be a sequence of numbers of the interval $[0, 1)$.

1) If $\sum_{k=1}^{\infty}(1-\lambda_k) < \infty$, then

$$\lim_{n\to\infty} \prod_{k=1}^{n} \lambda_k > 0 \ .$$

2) If $\sum_{k=1}^{\infty}(1-\lambda_k) = \infty$, then

$$\lim_{n\to\infty} \prod_{k=1}^{n} \lambda_k = 0 \ .$$

The proof of the lemma is offered in Appendix.

From the above lemma and Eq. (18), the convergence theorem 4.3 can be generalized further as follows.

Theorem 4.5: Given any initial conditions, assume that a consensus solution $\tilde{x}$ is found by a basic cooperative optimization algorithm at some iteration k and remains the same in the following iterations. If the series

$$(1-\lambda_1) + (1-\lambda_2) + \ldots + (1-\lambda_k) + \ldots \ , \tag{19}$$

is divergent, then

$$E(\tilde{x}) = E^*.$$

That is, the consensus solution $\tilde{x}$ must be the global minimal solution x^*, $\tilde{x} = x^*$.

If $1 - \lambda_k = 1/k$, for instance, the series (19) is the harmonic series,

$$1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} + \ldots$$

The harmonic series is divergent. Hence, with the choice of $\lambda_k = 1 - 1/k$, if a consensus solution $\tilde{x}$ is found at some iteration and it remains the same in the following iterations, it must be the global minimal solution x^*.

If $\{1 - \lambda_k\}_{k \geq 1}$, as another example, is a convergent sequence of a positive limit, then $\sum_k 1 - \lambda_k$ is divergent. In this case, a consensus solution is also the global minimal solution. This statement can be generalized further to Cauchy sequences. Every convergent sequence is a Cauchy sequence, and every Cauchy sequence is bounded. Thus, if $\{1 - \lambda_k\}$ is a Cauchy sequence of a positive bound, a consensus solution is the global minimal solution.

To maximize the performance of a cooperative optimization algorithm, it is popular in the experiments to progressively increase the cooperation strength as the iteration of the algorithm proceeds. A weak cooperation level at the beginning leads to a fast convergence rate (see Theorem 4.2). A strong cooperation level at a later stage of the iterations increases the chance of finding a consensus solution. Theorem 4.5 offers us some general guidance and justification for choosing a variable cooperation strength. It tells us that the increment of the cooperative strength should not be too fast if we want the guarantee of a consensus solution being the global optimal one.

V. General Canonical Form of Cooperative Optimization

By combining different forms of lower bound functions and different ways of decomposing objective functions, we can design cooperative optimization algorithms of different complexities and powers for attacking different optimization problems. The basic canonical form of cooperative optimization (7) can be generalized further in a straightforward way to the general canonical one as follows.

Given a multivariate objective function $E(x_1, x_2, \ldots, x_n)$ of n variables, or simply denoted as $E(x)$, where each variable is of a finite domain. Assume that $E(x)$ can be decomposed into m sub-objective functions $E_i(x)$ which may satisfy the condition

$$E(x) = \sum_{i=1}^{m} E_i(x) \ .$$

One may define another function $E_-(x)$, on the same set of variables as $E(x)$, as the composition of m component functions as follows,

$$E_-(x) = \sum_{i=1}^{m} \Psi_i(x^i) \ ,$$

where $\Psi_i(x^i)$ is a component function defined on a subset of variables $X_i^{'}$, $X_i^{'} \subset \{x_1, x_2, \ldots, x_n\}$, for $i = 1, 2, \ldots, m$. x^i is the restriction of x on $X_i^{'}$, denoted as $x^i = x(X_i^{'})$.

A cooperative optimization algorithm of the general canonical form is defined as minimizing the m sub-objective functions $E_i(x)$ in the following iterative and cooperative way,

$$\Psi_i^{(k)}(x^i) = \min_{X_i \setminus X_i^{'}} \left(E_i(x) + \lambda_k \sum_{j=1}^{m} w_{ij} \Psi_j^{(k-1)}(x^j) \right) , \tag{20}$$

for $i = 1, 2, \ldots, m$. In the equations, $k (= 1, 2, 3, \ldots)$ is the iteration step; X_i is the set of variables contained in the functions at the right side of the i-th equation; λ_k is a real value parameter at iteration k satisfying $\lambda_k \geq 0$; and w_{ij} $(1 \leq i, j \leq m)$ are also real value parameters satisfying $w_{ij} \geq 0$.

The solution at iteration k is defined as

$$\tilde{x}^{(k)} = \arg\min_{x} E_{-}^{(k)}(x) \ .$$

Moreover, $\tilde{x}^{(k)}$ is called a consensus solution if it is the conditional optimum of all the m minimization problems defined in (20). That is,

$$\tilde{x}^{(k)}(X_i) = \arg\min_{x(X_i)} \left(E_i(x) + \lambda_k \sum_{j=1}^{m} w_{ij} \Psi_j^{(k-1)}(x^j) \right) ,$$

when $x(X_i^{'}) = \tilde{x}^{(k)}(X_i^{'})$ and $i = 1, 2, \ldots, m$.

One may choose the parameters λ_k and w_{ij} in such a way that they further satisfy the conditions of $\sum_i w_{ij} = 1$, for all js, and all λ_ks are less than one ($\lambda_k < 1$). With the settings, if the algorithm reaches its equilibrium at some iteration and the solution of the iteration is also a consensus one, then it must be the global minimal solution (This global optimality condition can be proved in the exact same way as that of Theorem 4.4).

The general canonical form can be further generalized to variable propagation matrices, variable forms of lower bound functions, and variable ways of decomposing objective functions.

VI. Design Issues

A basic cooperative optimization algorithm (3) (or a basic canonical one (7)) is uniquely defined by the objective function decomposition $\{E_i(x)\}$, the cooperation strength series $\{\lambda_k\}_{k \geq 1}$, and the propagation matrix $(w_{ij})_{n \times n}$. Some general guideline for designing the cooperation strength series has discussed in the previous section. This section focuses on the rest two.

A. *Objective Function Decomposition*

1) Constraint Optimization Problems: A large class of optimization problems have objective functions of the following form,

$$E(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} f_i(x_i) + \sum_{(i,j)\in\mathcal{N}} f_{ij}(x_i, x_j) \ . \tag{21}$$

The function $f_i(x_i)$ is an unary function on variable x_i, for $i = 1, 2, \ldots, n$. and the function $f_{ij}(x_i, x_j)$ is a binary function on two variables x_i, x_j. To note the collection of all defined binary functions, the set $\mathcal{N}$ is used which contains non-ordered pairs of variable indices where each pair (i, j) corresponds to a defined binary function $f_{ij}(x_i, x_j)$.

The above optimization problems are also referred to as binary constraint optimization problems (binary COP) in AI. The unary function $f_i(x)$ is called an unary constraint on variable x_i and the binary function $f_{ij}(x_i, x_j)$ is called a binary constraint on variables x_i, x_j.

Binary constraint optimization problems are a very general formulation for many optimization problems arose from widely different fields. Examples are the famous traveling salesman problems, weighted maximum satisfiability problems, quadratic variable assignment problems, stereo vision, image segmentation, and many more. Solving a binary constraint optimization problem is NP-hard in computation.

2) Graphical Representation of Objective Functions: An objective function in form of (21) can be represented with an undirected graph $G = (V, E)$. In the graph, each variable x_i is represented by a node, called a variable node, $V = \{x_1, \ldots, x_n\}$; each binary constraint $f_{i,j}(x_i, x_j)$ is represented by an undirected edge, connecting the variable nodes x_i and x_j, denoted by a non-ordered pair of variable nodes (x_i, x_j). By definition, the set E of the edges of the graph G is $E = \{(x_i, x_j)|(i, j) \in \mathcal{N}\}$.

The simple example described in subsection II-E is a binary constraint optimization problem. The objective function (5) of the simple example has the form of (21). It can be represented by an undirected graph as shown in Figure 4.

If edge $(x_i, x_j) \in E$, then the variable nodes x_i, x_j are called neighbors to each other. In graph theory, they are also called adjacent to each other. Each variable node x_i can have a number of neighboring variable nodes. Let $\mathcal{N}(i)$ be the set of the indices of the neighboring variables of x_i. By definition,

$$\mathcal{N}(i) = \{j|(i, j) \in \mathcal{N}\} \ .$$

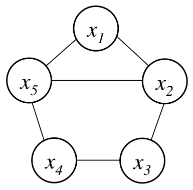


Fig. 4. The graphical representation of the objective function of the simple example.

Using the notations, we can rewrite the objective function (21) as

$$E(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \left(f_i(x_i) + 1/2 \sum_{j \in \mathcal{N}(i)} f_{ij}(x_i, x_j) \right) . \tag{22}$$

3) Straightforward Decompositions: The expression (22) for an objective function of a binary constraint optimization problem also defines a straightforward way to decompose the energy function. That is,

$$E_i(x) = f_i(x_i) + 1/2 \sum_{j \in \mathcal{N}(i)} f_{ij}(x_i, x_j), \qquad \text{for } i = 1, 2, \ldots, n \ . \tag{23}$$

Obviously, $\sum_{i=1}^{n} E_i(x) = E(x)$. This kind of decompositions is so called the straightforward decompositions. The sub-objective functions $E_i(x)$ in the straightforward decompositions can be easily minimized as

$$\min_x E_i(x) = \min_{x_i} \left(f_i(x_i) + 1/2 \sum_{j \in \mathcal{N}(i)} \min_{x_j} f_{ij}(x_i, x_j) \right) \ .$$

Using the graphical representation of an objective function can help us to visualize the straightforward decomposition of an objective function. For example, the decomposition of the objective function of the simple example presented in Subsection II-E can be viewed an instance of this kind of decompositions. The original objective function has a graphical representation shown in Figure 4. Each sub-objective function $E_i(x)$ of the decomposition can also be represented by a graph, which must be a subgraph of the original one. The graphical representation of the decomposition is illustrated in Figure 5. In the figure we can see that the original loopy graph is decomposed into five loop-free subgraphs.

In general, in a graphical representation, the straightforward decompositions given in (23) can be understood as decomposing a loopy graph into n loop-free subgraphs, one subgraph is

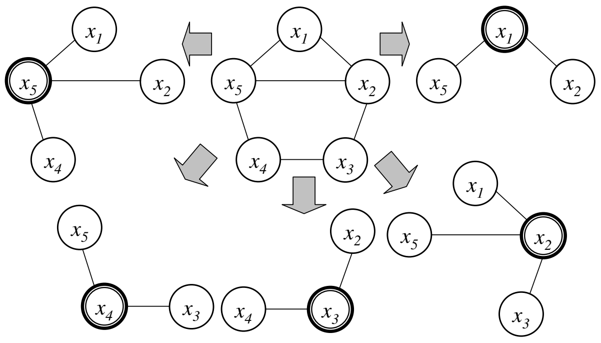


Fig. 5. Decomposing a loopy graph into a number of loop-free subgraphs based on the straightforward decompositions, one subgraph associated with each variable node (double circled).

associated with each variable node in the original graph. In the decomposition, each subgraph is of a star-like structure with its associated variable node as the star center. It consists of the variable node, the neighbors of the node, and the edges connecting the node with its neighbors.

4) Graph-Based Decompositions: The graphical representation of an objective function may contain many loops. That is the major cause of the difficulty at minimizing the objective function. If the graph is loop-free, there exist algorithms with linear complexity (e.g., dynamic programming) that can minimize the objective function efficiently. Therefore, if we can decompose an objective function with a loopy graphical representation into a number of sub-objective functions with loop-free graphical representations, a hard optimization problem is, thus, broken into a number of sub-problems of lower complexities. Cooperative optimization can then be applied to solve those sub-problems in a cooperative way. This kind of decompositions is called the graph-based decompositions.

It is important to note that the modification given in (2) for a sub-objective function does not change its graphical structure. In other words, every modified sub-objective function defined in (2) has the exact same graphical structure as its original one. This is because only unary functions, the assignment constraints $\Psi_i^{(k-1)}(x_i)$, are introduced in the definition. Therefore, any optimization algorithm applicable to the original sub-objective functions should also be applicable to the modified ones. In other words, if a sub-objective function is of a tree-like structure, then its modified version defined by (2) must have the exact same tree-like structure. Both of them can be minimized efficiently via dynamic programming.

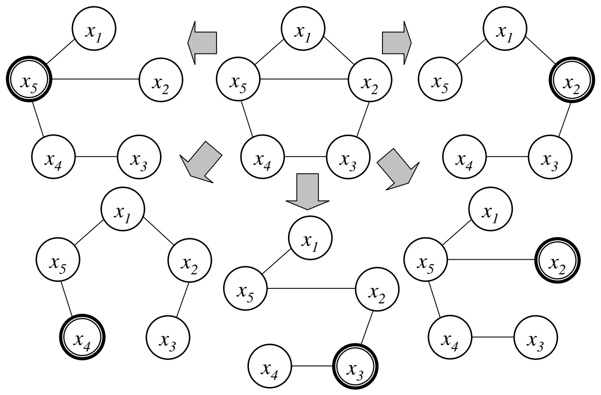


Fig. 6. Decomposing a loopy graph into a number of spanning trees, one spanning tree associated with each variable node (double circled).

5) Spanning-Tree Based Decompositions: In terms of the graph based decompositions, the straightforward decompositions are based on the direct neighbors of each variable node. Another possible way of decomposing an objective function is based on the spanning trees of the graph representing the function. A tree is called a spanning tree of an undirected graph G if it is a subgraph of G and containing all vertices of G. Every finite connected graph G contains at least one spanning tree T.

Given an objective function $E(x)$ of n variables in form of (21). Let $G = (V, E)$ be its graphical representation with n variable nodes. Without loss of generality, we assume that G is connected (otherwise it implies that the original minimization problem can be broken into several independent sub-problems). For each variable node x_i of G, we can associate a spanning tree of G, denoted as $T_i = (V, E_i)$ (T_i shares the same set of nodes as G). There are n such spanning trees in total, $T_1, T_2, \ldots, T_n$ (some trees may possibly be duplicated). We also choose those n spanning trees in a way such that each edge of G is covered at least by one of the n trees. Figure 6 shows an example of decomposing the graph representing the objective function of the simple example into five spanning trees.

After decomposing the graph G into n spanning trees, if we can define a sub-objective function $E_i(x)$ for each spanning tree such that

1) $E(x) = \sum_i E_i(x)$;
2) The graphical representation of $E_i(x)$ is T_i, for $i = 1, 2, \ldots, n$.

Then the set $\{E_i(x)\}$ is a legitimate decomposition of the original objective function $E(x)$. This kind of decompositions is so called the spanning-tree based decompositions.

Given an objective function $E(x)$ of a binary constraint optimization problem with a graphical representation G and n spanning trees, each unary constraint $f_j(x_j)$ of $E(x)$ is associated with a variable node x_j in G covered by all the n spanning trees. Each binary constraint $f_{jk}(x_j, x_k)$ of $E(x)$ is associated with an edge (x_j, x_k) in G covered at least by one of the n spanning trees. Assume that the edge $(x_j, x_k) \in E$ is covered by $m(j,k)$ spanning trees, where $m(j,k)$ should be a positive integer. One way for defining sub-objective functions $E_i(x)$ to satisfy the above two conditions is given as follows,

$$E_i(x) = \frac{1}{n}\sum_{j=1}^{n} f_j(x_j) + \sum_{(j,k)\in E_i} \frac{1}{m(j,k)} f_{jk}(x_j, x_k)\ , \tag{24}$$

for $i = 1, 2, \ldots, n$.

6) Some Properties of Spanning-Tree Based Decomposition: We can apply algebraic graph theory to reveal some of properties of graphs and their spanning trees. Let $G = (V, E)$ be a finite simple graph of n nodes, with vertices $v_1, \ldots, v_n$. If G is a graphical representation of an objective function, then the vertices v_i is the variable node x_i, i.e., $v_i = x_i$.

The connectivity of a graph G can be represented by the adjacency matrix A of G. The adjacency matrix A of G is defined as

$$A = (a_{ij})_{n\times n} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E, \\ 0, & \text{if } (v_i, v_j) \notin E. \end{cases}$$

The adjacency matrix of an undirected graph is symmetric.

Let Q be the adjacency matrix of G where the diagonal entries Q_{ii} are replaced by the degrees of vertices $-deg(v_i)$. Let Q^* be the matrix obtained by removing the first row and column of Q. Then the number of spanning trees in G is equal to $|det(Q^*)|$ (Kirchoff's Matrix-Tree Theorem). Particularly, if G is a complete graph, Q^* has the determinant n^{n-2}. That is, every complete graph with n vertices ($n > 1$) has exactly n^{n-2} spanning trees (Theorem of Cayley).

7) Further Generalization of Graph-Based Decompositions: Using factor graph [22], any objective function containing k-ary constraints, where k can be any integer number, can be represented as a graph. With the representation, the two aforementioned kinds of decompositions can be easily generalized further to decompose the objective function. Special decompositions can also be explored for graphs with special structures or constraints of some special properties to maximize the power of cooperative optimization algorithms. Another way to apply the two kinds of decompositions for the k-ary constraints case is by converting the constraints of orders higher than two into binary constraints via variable clustering technique.

B. The Propagation Matrix

As mentioned in the previous subsection, the objective function of a binary constraint optimization problem has a graphical representation $G = (V, E)$. For the straightforward decompositions described in the previous subsection, we can design a propagation matrix based on the adjacency of the graph G as follows

$$W = (w_{ij})_{n\times n} = \begin{cases} 1/d_j, & \text{if } (x_i, x_j) \in E, \\ 0, & \text{otherwise}. \end{cases} \tag{25}$$

where d_j is the degree of the variable node x_j. The propagation matrix (6) of the simple example is designed in this way.

Another way to design a propagation matrix is given as follows,

$$W = (w_{ij})_{n\times n} = \begin{cases} 1/(d_j+1), & \text{if } (x_i, x_j) \in E \text{ or } i = j, \\ 0, & \text{otherwise}. \end{cases} \tag{26}$$

Such a matrix has all of the diagonal elements of non-zero values.

C. Cooperative Optimization in Simple Form

The design of cooperative optimization algorithms is not trivial even with the aforementioned guidelines. In the basic canonical form (7), there are $n \times n$ values for the propagation matrix $(w_{ij})_{n\times n}$ and a series of values for the cooperation strength λ_k. To ease the design job for engineers and practitioners, the difference equations (7) of the basic canonical form can be simplified to

$$\Psi_i^{(k)}(x_i) = \min_{X_i \setminus x_i} \left(E_i(x) + \alpha \sum_j \Psi_j^{(k-1)}(x_j) \right), \quad \text{for } i = 1, 2, \ldots, n \ , \tag{27}$$

where α is the only parameter to be tuned in experiments to maximize performance. It plays the same role as the cooperation strength λ_k for controlling the cooperation level among the agents in a system. The above set of simplified difference equations defines the simple form of cooperative optimization.

The simple form is derived from the basic canonical form (7) by setting w_{ij} be a positive constant w, for any i and j, if x_j is contained in $E_i(x)$; and $w_{ij} = 0$, otherwise. We also let the cooperation strength λ_k be of a constant value λ. Let $\alpha = \lambda w$, we have (7) simplified to (27).

If the parameter α is of a large value, the difference equations (27) of a simple cooperative optimization algorithm may have value overflow problems in computing the assignment constraints $\Psi_i^{(k)}(x_i)$. To improve its convergence property, we can offset each $\Psi_i^{(k)}(x_i)$ by a value at each iteration. One choice is the minimal value of $\Psi_i^{(k)}(x_i)$. That is we offset $\Psi_i^{(k)}(x_i)$ by its minimal value as follows,

$$\Psi_i^{(k)}(x_i) := \Psi_i^{(k)}(x_i) - \min_{x_i} \Psi_i^{(k)}(x_i) \ .$$

Thus, the offsetting defines an operator on $\Psi_i^{(k)}(x_i)$, denoted as $\mathcal{O}(\Psi_i^{(k)}(x_i))$. With the notation, the difference equations of a simple cooperative optimization algorithm become

$$\Psi_i^{(k)}(x_i) = \mathcal{O}\left(\min_{X_i \setminus x_i}\left(E_i(x) + \alpha \sum_j \Psi_j^{(k-1)}(x_j)\right)\right), \quad \text{for } i = 1, 2, \ldots, n \ . \tag{28}$$

VII. A Case Study in Computer Vision

Just like many other problems in computer vision and image processing, stereo matching can be formulated as a binary constraint optimization problem with an objective function $E(x)$ in form of (21). For detail about the energy function definitions used for stereo matching, please see [13]. Basically, an unary constraint $f_i(x_i)$ in (21) measures the difference of the intensities between site i from one image and its corresponding site in another image given the depth of the site. A binary constraint $f_{ij}(x_i, x_j)$ measures the difference of the depths between site i and site j. This type of constraints is also referred to as the smoothness constraint in literature. It has also been widely used in solving image segmentation and other vision tasks.

In our experiments, we apply the simplified form of cooperative optimization (28) for stereo matching with the parameter α is set to 0.16. The maximum number of iterations is set to 16. The objective function associated with stereo-matching is decomposed based on the spanning-tree based decomposition. The detail of the decomposition and the minimization of the sub-objective functions are offered in the following subsection.

A. Decomposing Grid-like Graphs

Often times, the graphical representation of the objective function of an image segmentation problem or a stereo matching problem is of a 2-D grid-like structure. Because a 2-D grid-like graph is highly regular in structure, its spanning trees can be easily defined in a systematic way.

Given an objective function of a 2-D grid-like graphical representation $G = (V, E)$, let M be the height of the grid (the number of rows) and N be the width of the grid (the number of columns). Let E^h be the set of all horizontal edges of G and E^v be the set of all vertical edges. There are in total $M \times N$ nodes, one for each variable. There are in total $M(N-1)$ horizontal edges and $N(M-1)$ vertical edges. With the notations, the objective function can be expressed as

$$E(x) = \sum_{i \in V} f_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j) \ ,$$

or equivalently,

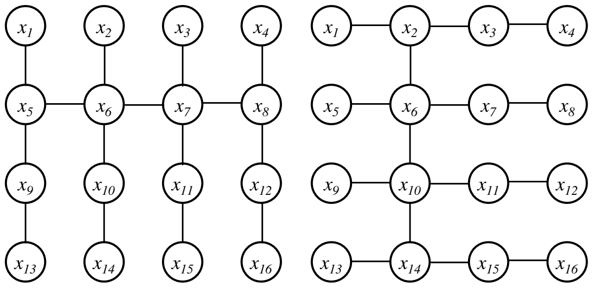
$$E(x) = \sum_{i \in V} f_i(x_i) + \sum_{(i,j) \in E^h} f_{ij}(x_i, x_j) + \sum_{(i,j) \in E^v} f_{ij}(x_i, x_j) \ .$$

The horizontal path $P_i^h = (V_i^h, E_i^h)$ through a variable node x_i consists of all the nodes at the same horizontal line as x_i, together with the edges connecting those nodes. The vertical path $P_i^h = (V_i^v, E_i^v)$ through a variable node x_i consists of all the nodes at the same vertical line as x_i, together with the edges connecting those nodes.

For each variable node x_i, let us define two spanning trees with the node as the root, called the horizontal spanning tree T_i^h and the vertical spanning tree T_i^v, respectively. The horizontal spanning tree T_i^h consists of the horizontal path through the variable node x_i and all the vertical paths through each node in the horizontal path. The vertical spanning tree T_i^v consists of the vertical path through the variable node x_i and all the horizontal paths through each node in the vertical path (the illustrations are shown in Fig. 7).

Let the functions $E_i^h(x), E_i^v(x)$ be the objective functions associated with the horizontal spanning tree T_i^h and the vertical spanning tree T_i^v, respectively. Following the general design guideline described in the previous section for the spanning-tree based decompositions (see Eq. (24)), we can define $E_i^h(x)$ and $E_i^v(x)$ as

$$E_i^h(x) = a \sum_{i' \in V} f_{i'}(x_{i'}) + b \sum_{(i',j) \in E_i^h} f_{i'j}(x_{i'}, x_j) + c \sum_{(i',j) \in E^v} f_{i'j}(x_{i'}, x_j) \ ,$$



x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}

Fig. 7. A horizontal spanning tree (left) and a vertical spanning tree (right) with the variable node x_6 as their roots.

$$E_i^v(x) = a \sum_{i' \in V} f_{i'}(x_{i'}) + c \sum_{(i',j) \in E_i^v} f_{i'j}(x_{i'}, x_j) + b \sum_{(i',j) \in E^h} f_{i'j}(x_{i'}, x_j) \ ,$$

where $a = 1/2MN$, $b = 1/(MN + N)$, and $c = 1/(MN + M)$.

The sub-objective function $E_i(x)$ associated with variable x_i is defined as

$$E_i(x) = E_i^h(x) + E_i^v(x) \ . \tag{29}$$

Clearly, we have $\sum_i E_i(x) = E(x)$.

As mentioned before, any objective function of a tree-like structure can be minimized efficiently using the dynamic programming technique. It is of a linear computational complexity and is simply based on local message passing from the leave nodes all the way back to the root node. The books [41], [23] offer a detail explanation about message passing and the dynamic programming technique. When applying the technique for minimizing the objective function of a horizontal or vertical spanning tree, the message flows among the variable nodes are illustrated in Figure 8.

B. Experimental Results

The Middlebury College evaluation framework [13] for stereo matching is used in the experiments. The script used for evaluation is based on *exp6_gc.txt* offered in the framework. The other settings come from the default values in the framework. The results of stereo matching algorithms together with the ground truths for the four test stereo image pairs from the evaluation framework are shown in Figure 9. The quality of solutions of both algorithms are very close to each other from a visual inspection.

$$\begin{tikzcd}
x_1 \arrow[d] & x_2 \arrow[d] & x_3 \arrow[d] & x_4 \arrow[d] & & x_1 \arrow[r] & x_2 \arrow[d] & x_3 \arrow[l] & x_4 \arrow[l] \\
x_5 \arrow[r] & x_6 & x_7 \arrow[l] & x_8 \arrow[l] & & x_5 \arrow[r] & x_6 & x_7 \arrow[l] & x_8 \arrow[l] \\
x_9 \arrow[u] & x_{10} \arrow[u] & x_{11} \arrow[u] & x_{12} \arrow[u] & & x_9 \arrow[r] & x_{10} \arrow[u] & x_{11} \arrow[l] & x_{12} \arrow[l] \\
x_{13} \arrow[u] & x_{14} \arrow[u] & x_{15} \arrow[u] & x_{16} \arrow[u] & & x_{13} \arrow[r] & x_{14} \arrow[u] & x_{15} \arrow[l] & x_{16} \arrow[l]
\end{tikzcd}$$

Fig. 8. The message flows among the variable nodes when applying the dynamic programming technique for minimizing the objective functions associated with the horizontal spanning tree (left) and the vertical spanning tree (right) with the variable node x_6 at their roots.

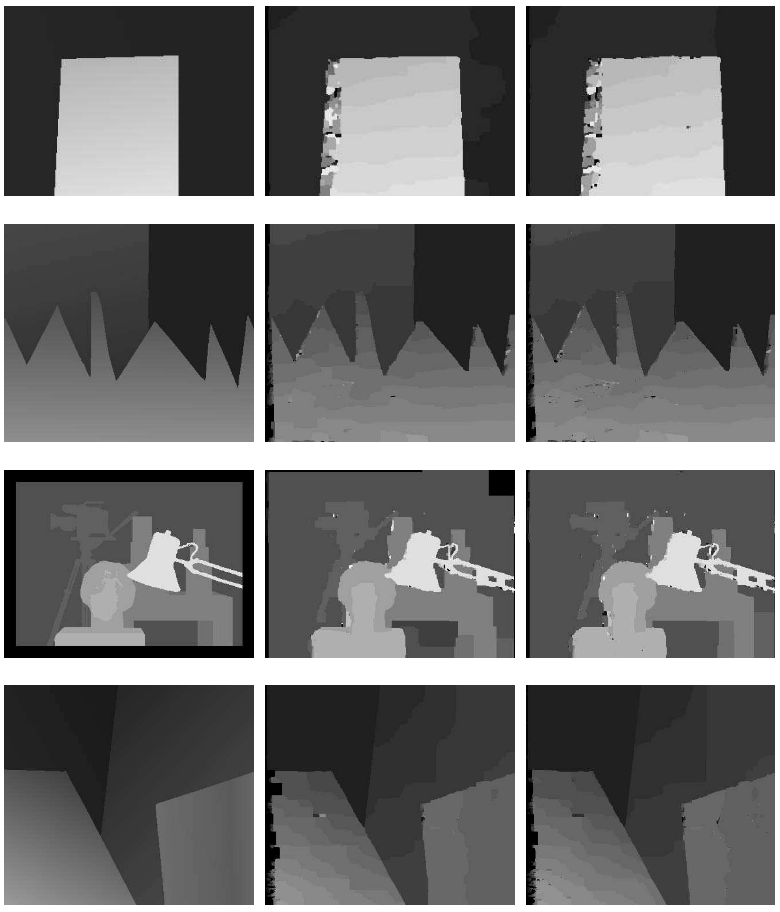


Fig. 9. The ground truths (the left column), cooperative optimization (the middle column), and graph cuts (the right column).

The following four tables show the performance of the cooperative optimization algorithm (upper rows in a table) and the graph cut algorithm (lower rows in a table) over the four test image sets. The solution quality is measured in the overall area, no occluded areas, occluded areas, textured areas, texture-less areas, and near discontinuity areas (see [13] for the detail description of the evaluation framework). Both algorithms does not handle occluded areas (an occluded area is one that is visible in one image, but not the other). Also, the runtimes of the two algorithms (co = cooperative optimization algorithm, gc = graph cuts) are listed. From the tables we can see that the cooperative optimization is very close to graph cuts in terms of solution quality and energy states. However, the cooperative optimization algorithm is around 20 times faster than graph cuts in the software simulation.

image = Map

time: co=17s / gc=337s energy: co=328,658 / gc=321,144

	ALL	NON OCCL	OCCL	TEXTRD	TEXTRLS	D_DISCNT
Error	4.04	0.85	16.18	0.85	0.36	2.84
Bad Pixels	5.35%	0.18%	86.78%	0.18%	0.00%	2.46%
Error	3.91	1.07	15.45	1.07	0.38	3.65
Bad Pixels	5.63%	0.36%	88.76%	0.36%	0.00%	4.52%

image = Sawtooth

time: co=33s / gc=673s energy: co=1,430,450 / gc=1,418,015

	ALL	NON OCCL	OCCL	TEXTRD	TEXTRLS	D_DISCNT
Error	1.46	0.61	7.92	0.63	0.33	1.56
Bad Pixels	3.93%	1.35%	93.06%	1.48%	0.14%	5.96%
Error	1.49	0.70	7.88	0.73	0.40	1.60
Bad Pixels	3.99%	1.38%	94.02%	1.49%	0.31%	6.39%

image = Tsukuba

time: co=20s / gc=476s energy: co=517,591 / gc=503,962

	ALL	NON OCCL	OCCL	TEXTRD	TEXTRLS	D_DISCNT
Error	1.30	0.99	5.41	1.00	0.97	2.01
Bad Pixels	4.77%	2.59%	87.38%	2.57%	2.61%	10.63%
Error	1.25	0.92	5.35	1.04	0.73	2.02
Bad Pixels	4.24%	2.04%	87.60%	2.77%	1.05%	10.00%

image = Venus						
time: co=35s / gc=573s			energy: co=1,253,764 / gc=1,246,078			
	ALL	NON OCCL	OCCL	TEXTRD	TEXTRLS	D_DISCNT
Error	1.58	1.11	8.29	0.93	1.42	1.49
Bad Pixels	3.29%	1.65%	90.72%	1.38%	2.20%	7.28%
Error	1.47	0.95	8.33	0.81	1.18	1.31
Bad Pixels	3.58%	1.93%	91.55%	1.56%	2.68%	6.84%

VIII. Conclusions

Cooperative optimization offers us a general, distributed optimization method for attacking hard optimization problems. Soft decision making, message passing, and solution compromising are three important techniques for achieving cooperation among agents in a cooperative optimization system. The global optimality property of consensus solutions offers an appealing reason for agents in a system to compromise their solutions so that conflicts in their solutions can be resolved. The insights we gained at studying cooperative optimization might help us to apply the cooperation principle to understand or solve more generic decision optimization problems arose from fields like neurosciences, business management, political management, and social sciences.

IX. Appendix

A. Proof of Theorem 4.4

Proof: Although the proof of the theorem is simple and straightforward, the property it reveals is important for cooperative optimization.

Since $\tilde{x}$ is a consensus solution, substitute it into (3) we have

$$\Psi_i^{(\infty)}(\tilde{x}_i) = (1-\lambda)E_i(\tilde{x}) + \lambda \sum_j w_{ij} \Psi_j^{(\infty)}(\tilde{x}_j),$$

for $1 \leq i \leq n$. Sum them up, we have

$$\begin{aligned} \sum_i \Psi_i^{(\infty)}(\tilde{x}_i) &= \sum_i \left((1-\lambda)E_i(\tilde{x}) + \lambda \sum_j w_{ij} \Psi_j^{(\infty)}(\tilde{x}_j) \right) \\ &= (1-\lambda)E(\tilde{x}) + \lambda \sum_j \Psi_j^{(\infty)}(\tilde{x}_j) \ . \end{aligned}$$

That is

$$E(\tilde{x}) = \sum_i \Psi_i^{(\infty)}(\tilde{x}_i) \ . \tag{30}$$

For any x, from (3), we have

$$\Psi_i^{(\infty)}(x_i) \leq (1-\lambda)E_i(x) + \lambda \sum_j w_{ij}\Psi_j^{(\infty)}(x_j),$$

for $1 \leq i \leq n$. Sum them up, we have

$$\begin{aligned}\sum_i \Psi_i^{(\infty)}(x_i) &\leq \sum_i \left((1-\lambda)E_i(x) + \lambda \sum_j w_{ij}\Psi_j^{(\infty)}(x_j)\right) \\ &= E(x) + \lambda \sum_j \Psi_j^{(\infty)}(x_j)\ .\end{aligned}$$

That is

$$E(x) \geq \sum_i \Psi_i^{(\infty)}(x_i)\ . \tag{31}$$

Subtract (30) from (31), we have

$$E(x) - E(\tilde{x}) \geq \sum_i \left(\Psi_i^{(\infty)}(x_i) - \sum_i \Psi_i^{(\infty)}(\tilde{x}_i)\right)\ . \tag{32}$$

Because

$$\Psi_i^{(\infty)}(x_i) \geq \Psi_i^{(\infty)}(\tilde{x}_i)\ ,$$

from (32), we have

$$E(x) - E(\tilde{x}) \geq 0\ .$$

Therefore, $\tilde{x}$ must be the global minimum of $E(x)$.

This completes the proof. § ■

B. Proof of Lemma 4.1

Proof: For any numbers $\lambda_1, \ldots, \lambda_k$ in $[0,1)$, the following inequality can be proved by the principle of mathematical induction,

$$\lambda_1\lambda_2\ldots\lambda_k \geq 1 - (1-\lambda_1) - (1-\lambda_2) - \ldots - (1-\lambda_k)\ .$$

If $\sum_{k=1}^{\infty}(1-\lambda_k)$ converges, there exists N such that for all $n \geq N$,

$$(1-\lambda_N) + \ldots + (1-\lambda_n) < \frac{1}{2}\ .$$

Therefore, defining $g(n)$ as

$$g(n) = \prod_{k=1}^{n} \lambda_k\ ,$$

we have that for all $n \geq N$,

$$\frac{g(n)}{g(N-1)} = \lambda_N \ldots \lambda_n \geq 1 - ((1-\lambda_N) + \ldots + (1-\lambda_n)) \geq \frac{1}{2} \ .$$

Therefore, the sequence $\{g(n)\}_{n \geq N}$ is a non increasing sequence bounded from below by $g(N-1)/2 > 0$. It must have a positive limit $\epsilon > 0$ so that

$$\lim_{n\to\infty} \prod_{k=1}^{n} \lambda_k = \epsilon > 0 \ .$$

Using the inequality $1 - x \leq e^{-x}$ when $x \in [0,1)$, we have that

$$g(n) = \prod_{k=1}^{n} (1 - (1-\lambda_k)) < e^{-((1-\lambda_1)+\ldots+(1-\lambda_n))} \ .$$

If $\sum_{k=1}^{\infty} (1-\lambda_k)$ is a divergent series, we have

$$\lim_{n\to\infty} \prod_{k=1}^{n} \lambda_k = 0 \ .$$

This completes the proof. § ■

References

[1] C. H. Papadimitriou and K. Steiglitz, Eds., *Combinatorial Optimization*. Dover Publications, Inc., 1998.

[2] P. Pardalos and M. Resende, *Handbook of Applied Optimization*. Oxford University Press, Inc., 2002.

[3] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*. New York: Springer-Verlag, 2002.

[4] Kirkpatrick, C. Gelatt, and M. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671–680, 1983.

[5] S. Geman and D. Geman, “Stochastic relaxation, gibbs distributions, and the bayesian restoration of images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[6] L. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: John Wiley, 1966.

[7] G. Hinton, T. Sejnowski, and D. Ackley, “Genetic algorithms,” *Cognitive Science*, pp. 66–72, July 1992.

[8] H.-P. Schwefel, *Evolution and Optimum Seeking*. John Wiley and Sons, Inc., 1995.

[9] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, Massachusetts, London, England: The MIT Press, 2004.

[10] F. Golver and M. Laguna, *Tabu Search*. Kluwer Academic Publishers, 1997.

[11] E. L. Lawler and D. E. Wood, “Brand-and-bound methods: A survey,” *OR*, vol. 14, pp. 699–719, 1966.

[12] E. G. C. Jr., Ed., *Computer and Job-Shop Scheduling*. New York: Wiley-Interscience, 1976.

[13] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *IJCV*, vol. 47, pp. 7–42, April-June 2002.

[14] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cut,” *IEEE TPAMI*, vol. 23, no. 11, pp. 1222–1239, 2001.

[15] C. L. Zitnick and T. Kanade, “A cooperative algorithm for stereo matching and occlusion detection,” *IEEE TPAMI*, vol. 2, no. 7, 2000.

[16] D. Marr and T. Poggio, “Cooperative computation of stereo disparity,” *Science*, vol. 194, pp. 209–236, June 1976.

[17] K. Atkinson, *Computers and Intractability*. San Francisco, U.S.A.: Kluwer Academic Publishers, 1989.

[18] A. Rosenfeld, R. Hummel, and S. Zucker, “Scene labelling by relaxation operations,” *IEEE Transactions on System, Man, and Cybernetics*, vol. SMC-6, no. 6, p. 420, June 1976.

[19] Y. Boykov and V. Kolmogorov, “An experiemental comparison of min-cut/max-flow algorithms for energy minimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, December 2004.

[20] V. Kolmogorov and R. Zabih, “What energy functions can be minimized via graph cuts?” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, February 2004.

[21] R. Szeliski and R. Zabih, “An experimental comparison of stereo algorithms,” in *Vision Algorithms: Theory and Practice*, ser. LNCS, B. Triggs, A. Zisserman, and R. Szeliski, Eds., no. 1883. Corfu, Greece: Springer-Verlag, Sept. 1999, pp. 1–19.

[22] F. R. Kschischang, B. J. Frey, and H. andrea Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, February 2001.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[24] N. Wiberg, “Codes and decoding on general graphs,” Ph.D. dissertation, Department of Electrical Engineering, Linkoping University, Linkoping, Sweden, 1996.

[25] S. M. Aji and R. J. McEliece, “The generalized distributive law,” *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 325–343, March 2000.

[26] M. F. Tappen and W. T. Freeman, “Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters,” in *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*. IEEE Computer Society, 2003, pp. 900–907.

[27] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, “Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches,” *IEEE Transactions on Information Theory*, vol. 51, no. 11, pp. 3697–3717, March 2005.

[28] V. Kolmogorov, “Convergent tree-reweighted message pasing for energy minimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568–1583, October 2006.

[29] X. Huang, “A cooperative search approach to global optimization,” *Proceeding of the First International Conference on Optimiztion Methods and Software*, vol. December, p. 140, Hangzhou, P.R. China, 2002.

[30] ——, “A general global optimization algorithm for energy minimization from stereo matching,” in *ACCV*, Korea, 2004, pp. 480–485.

[31] ——, “Solving combinatorial optimization problems from computer vision using recurrent neural networks,” *Proceedings of ICANN/ICONIP 2003*, pp. 282–285, June 2003.

[32] ——, “A general framework for constructing cooperative global optimization algorithms,” in *Frontiers in Global Optimization*, ser. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, 2004, pp. 179–221.

[33] ——, “Cooperative optimization for solving large scale combinatorial problems,” in *Theory and Algorithms for Cooperative Systems*, ser. Series on Computers and Operations Research. World Scientific, 2004, pp. 117–156.

[34] ——, “Cooperative optimization for energy minimization in computer vision: A case study of stereo matching,” in *Pattern Recognition, 26th DAGM Symposium*. Springer-Verlag, LNCS 3175, 2004, pp. 302–309.

[35] ——, “Near perfect decoding of LDPC codes,” in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, 2005, pp. 302–306.

[36] ——, "Fast min-sum algorithms for decoding of LDPC over $GF(q)$," in *IEEE Information Theory Workshop*. Chengdu, China: IEEE Press, 2006, pp. 96–99.

[37] ——, "Deriving the normalized min-sum algorithm from cooperative optimization," in *IEEE Information Theory Workshop*. Chengdu, China: IEEE Press, 2006, pp. 204–208.

[38] ——, "Image segmentation by cooperative optimization," in *IEEE International Conference on Image Processing (ICIP)*, Singapore, 2004, pp. 945–948.

[39] J. Chen and M. Fossorier, "Density evolution of two improved BP-based algorithms for LDPC decoding," *IEEE Communications Letters*, vol. 6, pp. 208–210, 2002.

[40] J. G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–78, Mar. 1973.

[41] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003, available from `http://www.inference.phy.cam.ac.uk/mackay/itila/`. [Online]. Available: http://www.cambridge.org/0521642981

[42] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, July 2005.

[43] M. Wainwright, T. Jaakkola, and A. Willsky, "A new class of upper bounds on the log partion function," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2313–2335, July 2005.

[44] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, February 2001.

[45] R. Hummel and S. Zucker, "On the foundations of relaxation labeling processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 3, p. 267, June 1983.

[46] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, pp. 99–118, 1977.

[47] A. Blake, "Comparison of the efficiency of deterministic and stochestic algorithms for visual reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-11, no. 1, pp. 2–12, January 1989.