Mathematical Problems of Computer Science 39, 135--145, 2013.

Adapting RGT Solver Interface to Management Strategy Search Problems

Tadevos S. Baghdasaryan

Institute for Informatics and Automation Problems of NAS of RA
e-mail: tbs_@mail.ru

Abstract

We develop a unified Solver for the class of problems where the space of possible solutions can be specified by reproducible game trees (RGT) [11].

In this paper we adapt the interface of RGT Solver for marketing and supply chain management (SCM) problems of the class presented by ValueWar and Trading Agents Competition models [1, 2, 4]. Particularly, we describe:

- the ways for construction of operating entities, actions and strategy plans for the ValueWar tool,
- the ways for construction of operating entities, actions, moves and strategy plans for the TAC SCM game.

Keywords: combating and competing games, expert systems, optimal strategies, ValueWar, TAC.

1. Introduction

1.1. Definition of RGT Class

In the variety of problems we identify the class where the space of possible solutions can be specified by Reproducible combinatorial Game Trees (RGT) and develop a unified software, RGT Solver, for elaborating optimal strategies for any input specified problem of the class [11].

As it was demonstrated in [11], RGT is a spacious class of problems with only a few following requirements to belong to:

- there are (a) interacting actors (players, competitors, etc. performing (b) identified types of actions in the (c) specified moments of time and (d) specified types of situations;

- there are identified benefits for each of the actors;

- the situations the actors act in and transformed after the actions, can be specified by certain rules, regularities.

We do solve *games* of *RGT* class with meanings we do specify by *states, situations, actors, actions* of players, *evaluators* of situations and *regularities* of transformation of situations [11].

Many security and competition problems belong to RGT class since those problems always interact, and RGT requirements include the most common of them. Specifically, these are network Intrusion Protection (IP), Management in oligopoly competitions and Chess-like combinatorial problems, various security problems. The unified RGT specification of problems makes possible to design a unified Solver for the problems of the class [9, 10]. The Solver of RGT problems is a package aimed to acquire strategic expert knowledge to become comparable with a human in solving hard combinatorial competing and combating problems.

1.2. RGT Interpretation of Management Strategy Search Problem

According to [11] market competitions by ValueWar [1-3] and SCM by Trading Agents Competitions [4] (TAC SCM) will be the following:

- states are determined by the set of parameters of current competition and scenarios of competition
- situations are determined by the states and actions of competition
- actors: a company competing against a few others
- actions are changes of the product price and quality in ValueWar case or operational moves in TAC game
- evaluators: algorithms calculating for input situations
- regularities, or transformation rules, in ValueWar case are determined by general micro- and macro- economics laws, which are applied to the situations. For TAC case these regularities are implemented on the TAC Server side.

The shell of RGT Solvers is developed to provide the user with friendly Java environment for unified solution of any RGT problem.

The aims of the paper are the following:

- to adapt marketing (by VW) and SCM (by TAC) problems into the framework of RGT Solver
- to develop ways for the construction of marketing and SCM strategies.

In the paper we present first the adaptation of ValueWar followed by TAC SCM.

ValueWar (VW) integration includes:

- presentation of its composing nucleous entities
- presentation of actions and strategy plans
- definitions of VW situations
- embedding VW interface into RGT Solver

TAC SCM integration includes:

- presentation of SCM nucleous and composite entities
- presentation of moves (actions) and strategy plans
- definition of situation in TAC SCM
- embedding TAC SCM interface into RGT Solver.

2. Adapting RGT Solver to ValueWar

2.1. ValueWar

So far the existing framework of RGT Solver was implemented for the chess problem. But it is being modified to be more flexible in order to include other problems of the RGT class as well, in particular case the marketing and supply chain management problems presented by ValueWar tool and TAC SCM model [9, 12, 13].

ValueWar is a tool for the marketing strategy analysis, which presents a model of oligopoly competition of a few companies competing within a specified market. Every company competes for one of the predefined goals. Each of them is assigned to one of the various Strategy Plans that describe qualitative changes of basic competition parameters - Price and Quality of items (services) they produce [2]. Running the simulation a number of times (every time with a different strategy plan) and having the needed results, it is possible to separate the most optimal strategy plan for the given competitor playing within the selected type of market [5].

2.2. ValueWar Nucleous Entities

As an equivalent of Figure in chess application of RGT solver, in ValueWar we represent operating parameters as Price (P) and Quality (Q).

ValueWar operates with strategy plans that represent the qualitive changes of the operating parameters mentioned above. In the frames of the Solver we describe each parameter as a nucleous element with a particular name, type and value range. Namely, for the price it would look in the following way [10]:

Price: name "Price" value range >=0	**Price** Price | > 0	Quality: name "Quality" value range in [0,100]	**Quality** Quality | > 0

2.3. Strategy Plans for ValueWar

Another equivalent is the definition of Action, which is represented as Move in chess and as a concept of Strategy Plan in ValueWar. An SP describes the manner of changing P and Q operational parameters. A typical example of SP would be, for instance, "Decrease P, Increase Q". At this point the actions are Increase and Decrease. Each action will change the operating parameter in a predefined manner and by some delta.

We describe Action objects with the Solver and further add their implementations within each reality. In our case we should apply Actions over Price and Quality, so the same action is implemented twice - both for the Price object and Quality object. Depending on the application the Agent selects and implements the corresponding version, passing the object as a parameter.

In other words, in real world we can create some set of verbs. And having some real objects we define how to apply a particular verb to a particular object. It is possible that some action cannot be applied to the given object. Such a case brings to nonsense unless we describe an appropriate implementation for that case.

Here we define some set of Actions (verbs) and some set of Objects (realities). So, for each object we have to create an implementation of the same Action and associations between the Object's instance of that verb and the original verb.

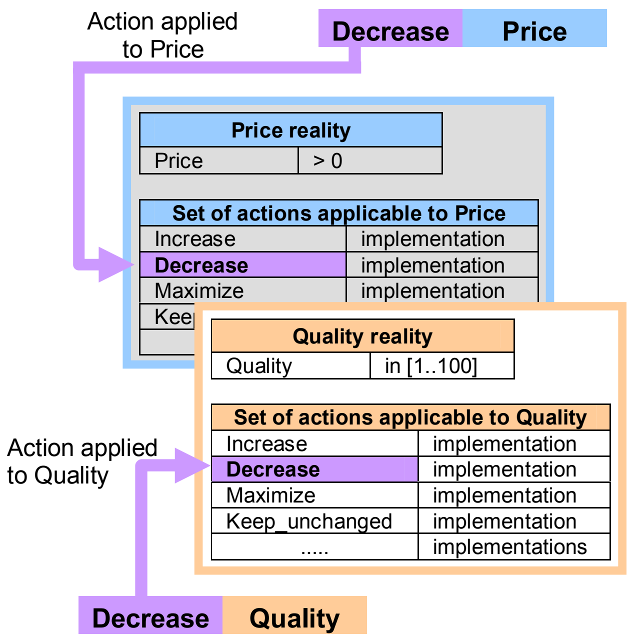


Fig. 1. Application of the same action to different entities: Price and Quality.

At the association level when met a term like “action over an object” (“Increase **P**rice”) the Agent starts to find a link between the “action” and its corresponding instance, related to the given “object”. The presence of such a link means that the action is applicable to that object and the Agent will implement that term, otherwise the term is interpreted as nonsense.

Thus, for example, “Increase” action will work differently for P and Q (assuming that it is implemented for both of them).

So, having such a set of actions, we create a set of various strategy plans and control the way of changing the operational parameters (P and Q).

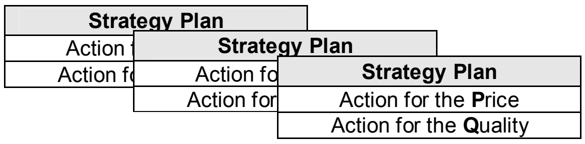


Figure 2. Set of ValueWar Strategy Plans.

2.4. Situation of ValueWar

A situation in ValueWar is considered as a current market state. All parameters that form the market model with their values in couple with parameters of the participating parties (competitors) represent a set that we will call a Situation (State). In its turn, every single move of a competitor (in accordance with its strategy plan) affects the market and changes its state, in other words, the competitor’s move changes the situation.

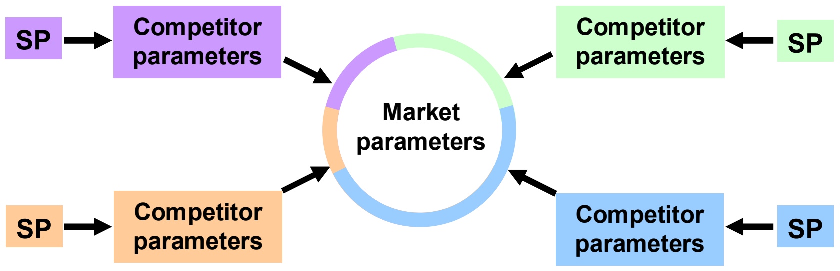


Fig. 3. Affection of competitors’ strategy plans on a ValueWar market model.

In order to run the created strategy plans (performing strategy plan simulation) in the frames of the RGT Solver we created a simple market model, based on all the parameters present in the basic ValueWar market model. Also, the market model may be switched to perform different predefined market types. In initial ValueWar there are about 70 parameters forming the model, including such factors as demand, supply, population, per-capita-income, sales, total sales, market share and others. Each of the forming factors is in interconnection with one or more others and is calculated by its corresponding formulas. The detailed description of the model is out of the scope of this paper and in fact, there may be different variations of it. Actually, any market model is acceptable for ValueWar framework problem, the only requirement is that the model environment should develop and change, affected by two single driving parameters - price and quality.

2.5. Embedding ValueWar Interface into RGT Solver

The ValueWar simulator is embedded into RGT Solver framework as a separate tab, which contains the main ValueWar gameboard and subwindows for displaying various parameters.

The gameboard is composed of two axes, forming a square of [P, Q] positions field. The competitors are presented on the board as smaller squares of predefined color and their positions on the board reflect their values of P and Q. As the simulation is in process, their positions may change in accordance to their strategy plans and achieved [P,Q] values.

Before the simulation starts, the user selects a type of a market and strategy plans for each of competing parties. Then the user may choose the simulation to proceed automatically or step-by-step. The simulation is splitted into N periods. During a single period each party makes one PQ-move and after that the market changes are calculated. Every party has its separate window that reflects its related market parameters. In addition, market has its own window for reflecting its specific factors' values. The user can change the strategy for any competitor at any period of the simulation, as well as the market type itself. In both cases the simulation continues with the new changes, starting from the next period.

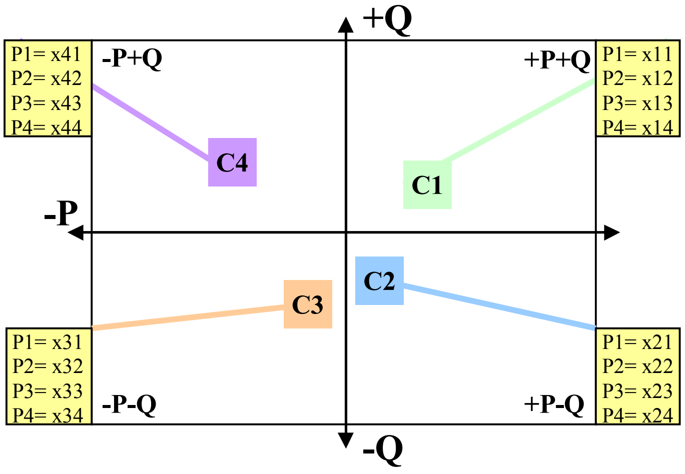


Figure 4. The ValueWar gameboard.

3. Adapting RGT Solver to TAC SCM

3.1. TAC SCM

Comparing to the ValueWar, TAC problem is more complex not only because of more parameters present, but also that in TAC there is higher level of uncertainty. There are two separate parts of a game – the first onedealing with customers and the second onedealing with suppliers [4].

In the TAC SCM game several agents compete in the market of personal computers assembling PCs of 16 configurations. Configurations depend on types of each 4 main supply components used: CPU, RAM, MotherBoard and HDD. Every main component is being produced by 2 supplier brands and available in 2 qualitative options: high and low (speed or capacity). Besides, there are limitations in compatibility of CPUs with MBoards: an MBoard of one brand must be equipped only by a CPU of the corresponding brand.

The actors in TAC SCM are:

- *Customers*, which compose the demand of different types of PCs. On the first *D* day of every TAC cycle they send numbers of RFQs (requests for quotes) to agents. An RFQ includes: a configuration of requested PC, quantity, desired price, delivery date and penalty for any delay.
- *Agents* (which have assembly Factories (with the given daily production capacity), Bank accounts and Warehouse), which receive RFQ bundles, analyze them and send back to customers as offer bundles (an offer includes: proposed configuration, quantity, price and delivery date). As a result, on the *D+1* second day of the cycle Customers may order some of the offers to the Agent.
- *Suppliers*, which produce the supply components. There are 2 Suppliers for each main component (CPU, HDD, RAM and MBoard). Each Supplier has its daily production capacity and may produce 2 qualities of the same supply component. Suppliers receive RFQ bundles

from Agents on day *D+1*, just after getting orders from Customers. An Agent RFQ includes: requested supply component, quantity, desired price and delivery date. On the next *D+2* day Suppliers send back offer bundles with available quantities and delivery dates of the requested supply to the Agent. Then the Agent may order some of those offers from the Supplier on the same day.

When an Agent gets all the necessary supplies needed for any particular Customer's order, it starts the assembling itself on its Factory. The assembling may last a number of days, taking into account: number of production cycles required for a single PC of the type, quantity of ordered PCs of the type and daily production capacity of the Factory, in cycles. After the order is completed it is delivered to the Customers and the Agent's Bank account is updated accordingly, subtracting penalties for any delay (if any). The minimal full TAC cycle lasts for 6 days.

The game duration is *T* periods (days). Every next day a new main TAC cycle starts, meanwhile the deals started in previous cycles are still being proceeded. After the last period the game is over and the agent with the biggest bank account is considered as a winner.

In ValueWar we had two driving parameters, P and Q – let's say two dimensions. In TAC they are much more: 4 types of components x 2 brands for each type x 2 qualities of each component, also various *dates*, etc – so, due to this "multidimentional factor" there is no explicit gameboard applicable.

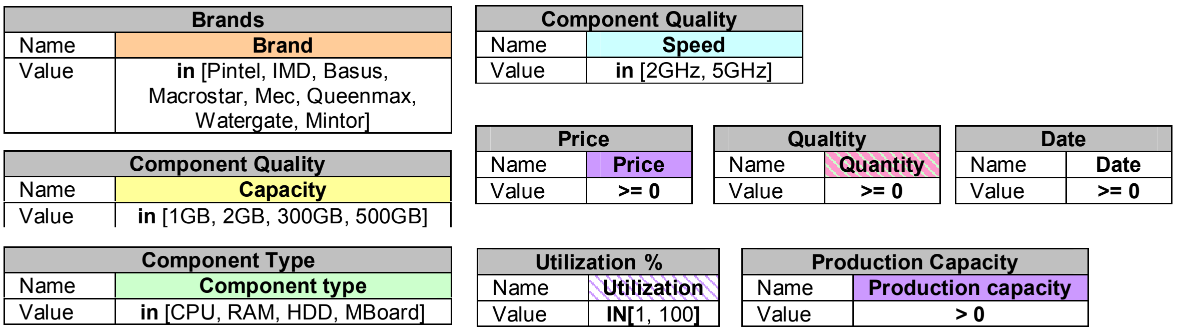
3.2. Nucleous and Composite Entities of TAC SCM

The main base nucleous entities for TAC are: Date, Price, Quantity, Speed, Base component type, Base component quality, Base component brand, Factory capacity, Factory utilization, Bank account, etc.

More complex (composite) entities are: Assembled product, Particular supply component, Bank, Factory, Agent, Customer, Supplier, etc. [10].

Let's take an example for component. There are 4 types of components: CPU, HDD, MBoard and RAM. Each of them is manufactured by 2 component-specific brands, and each brand produces 2 qualities of that component. For CPU component there are 2 manufacturers, Pintel and IMD, which manufacture 2 qualities of CPU - 5GHz and 2GHz. For HDD component there are 2 manufacturers (Watergate and Mintor) producing 2 qualities of them (300GB and 500GB). For RAM component Mec and Quinmax produce 2 qualities of them (1GB and 2GB). Finally, for MotherBoard component Basus and Macrostar produce motherboards for Pintel and IMD (there is a limitation that Pintel CPUs will work only on Pintel motherboards and IMD CPUs will work only on IMD motherboards).

First of all we should describe the basic nucleous elements with Solver (examples below describe Brands, Capacity, Component type, MBoard type, Price, Date, Utilization, Production capacity entities):



Brands	
Name	**Brand**
Value	**in** [Pintel, IMD, Basus, Macrostar, Mec, Queenmax, Watergate, Mintor]

Component Quality	
Name	**Speed**
Value	**in** [2GHz, 5GHz]

Component Quality	
Name	**Capacity**
Value	**in** [1GB, 2GB, 300GB, 500GB]

Price	
Name	**Price**
Value	**>= 0**

Qualtity	
Name	**Quantity**
Value	**>= 0**

Date	
Name	**Date**
Value	**>= 0**

Component Type	
Name	**Component type**
Value	**in** [CPU, RAM, HDD, MBoard]

Utilization %	
Name	**Utilization**
Value	**IN**[1, 100]

Production Capacity	
Name	**Production capacity**
Value	**> 0**

Fig. 5. Example TAC nucleous elements.

Then based on them, we can describe some specific elements (examples for capacities for different components, production-specific brands, types of Motherboards):

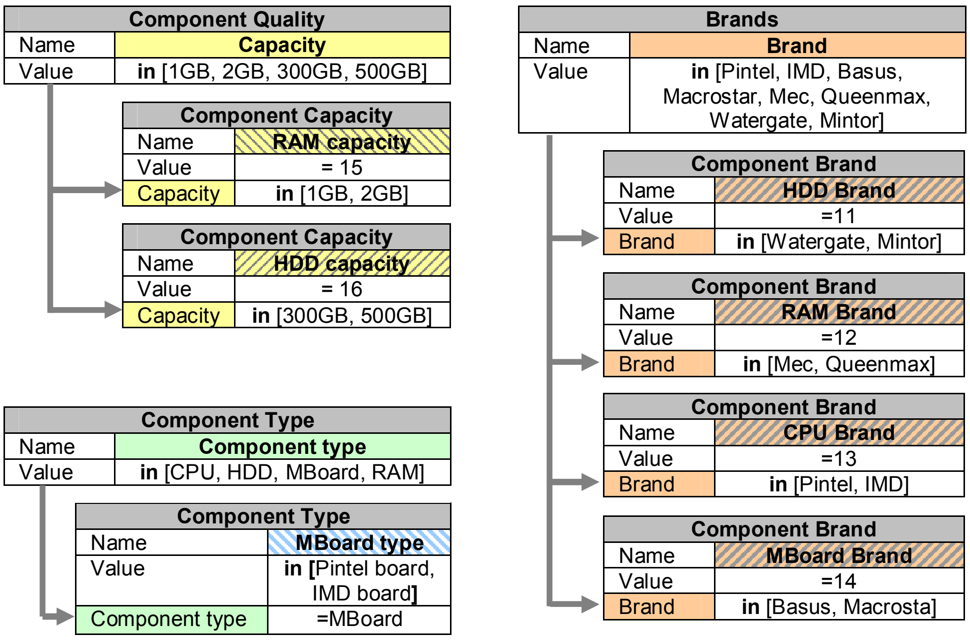


Fig. 6. Example of specific TAC nucleous elements.

Further, more complex and final entities will be described (particular supply components, like "IMD CPU 2GHz with base price of $1000", "Mintor HDD 500GB with base price of $300" or final entities like "CPU Supplier #1 branded 'Pintel', with production capacity of 2000 cycles"):

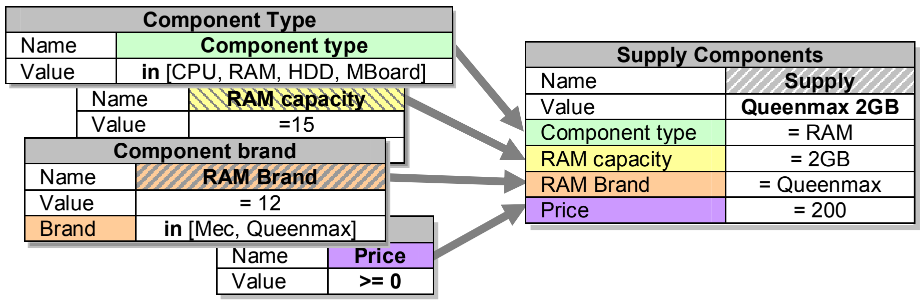


Fig.7. A complex TAC entity "Queenmax 2GB RAM" as a supply component

Following this path, we can describe all the entities present in the TAC problem, which will be equivalent to "figure" concept in a chess problem.

Below in Figure 8 some examples are given:

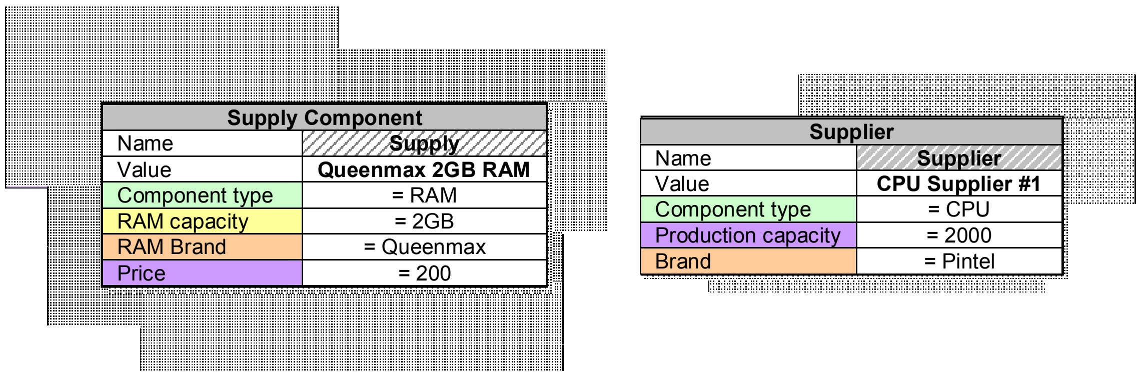


Fig. 8. Examples of TAC entities.

3.3. Defining Moves and Strategy Plans for TAC SCM

As in ValueWar, Strategy Plans for TAC specify the qualitative changes of some operational parameter(s). In the frames of TAC problem, we can separate two concepts from each other: Move and Strategy Plan (SP). We call a statement as "Move" which contains entities with all their parameters specified with exact values (equivalent to "figure move" in chess). In contrast, we call a statement as "SP" in which at least one parameter is not specified with its value. Again, an SP describes qualitative action(s) over TAC entity (entities), whereas a TAC move describes an exact (quantitative) action.

In case of move the Agent directly executes the statement.

In case of SP (with missing values of one or more parameters) the Agent may generate several moves from the same SP by substituting the missing values with some others from their acceptable value range. In order to select the most optimal move from the generated set it applies a game subtree for that appeared intermediate problem with possible usage of the existing knowledge base.

A typical example of TAC move would be the following statement: "Buy N quantity of Queenmax 2GB RAM by D date" [7]. Here we have 3 predefined entities: "Quantity", "Queenmax 2GB RAM" supply component and "Date". All three entities are populated with their exact values: Quantity=N, Date=D, Supply="Queenmax 2GB RAM".

Qualtity	
Quantity	N

Supply Component	
Supply	Queenmax 2GB RAM
Component type	= RAM
RAM capacity	= 2GB
RAM Brand	= Queenmax
Price	= 200

Date	
Date	D

Fig.8. Entities with fully populated values from the Move's statement.

Replacing at least a single value from any of parameters will force to interpret the given statement as a strategy plan as shown in examples:

Qualtity	
Quantity	■

Supply Component	
Supply	Queenmax 2GB RAM
Component type	= RAM
RAM capacity	
RAM Brand	= Queenmax
Price	= 200

Date	
Date	D

Qualtity	
Quantity	N

Supply Component	
Supply	Queenmax 2GB RAM
Component type	= RAM
RAM capacity	■
RAM Brand	= Queenmax
Price	= 200

Date	
Date	D

Fig.9. A single missing parameter in Quantity (above) or "Supply Component" (below) causes the statement to be treated as an SP.

Note, that in complex entities not just any parameter may be missed. If in the example above we replace the value of "Component type" ("RAM"), then the statement loses its meaning in TAC context: in "Buy N quantity of Queenmax 2GB by D date" the Agent will not understand the meaning of "Queenmax 2GB" – that is RAM, HDD, MBoard or CPU. Otherwise, it will require more complex searching mechanisms in the Agent implementation, which is currently out of scopeof this project.

Therefore, a typical example of SP would be the following statements:

- "Buy N quantity of Queenmax by D date" (as Queenmax produces RAMs of different capacity);
- "Buy quantity of RAM by D date"(Quantity may be any number within its predefined range);

- "Buy N quantity of 2GB RAM in advance" (Date is specified to be any coming number of days within the acceptable range);
- etc.

Also in the example above there is an action – "Buy". As in ValueWar, here in TAC we also should describe action objects with a further addition of their implementations into each entity it may be applied to. In the case above the "Buy" action should be described in general and its appropriate implementation should exist for "Queennmax 2GB RAM" entity. That action may also take "Quality" and "Date" entities as mandatory (the first) or optional (the second) parameters. If no "Buy" action is defined for "Queennmax 2GB RAM" entity, then that statement loses its meaning and that move or SP will take no effect. Note, that "Buy" is not defined either for entities "Quality" or "Date", so the separate statements "Buy *N* Quantity" or "Buy *D* Date" are senseless and will not work.

Generally, in real world we can create some vocabulary of verbs. Having some real objects we define how to apply a particular verb to a particular object. It is possible that some action cannot be applied to the given object. Such a case brings to nonsense unless we describe an appropriate implementation for that case.

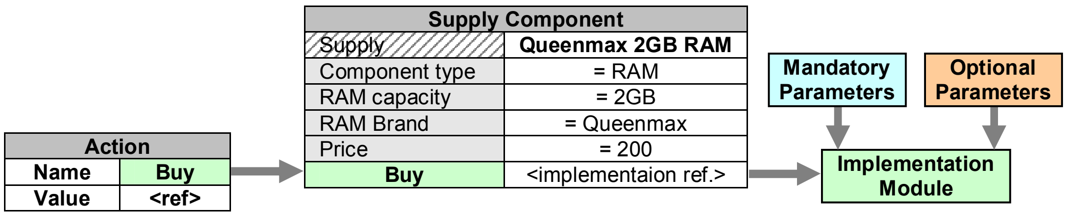


Fig. 10. Implementation of "*Buy* Queenmax 2GB RAM" action.

Here we define some set of Actions (verbs) and some set of Objects (realities). So, for each object we have to create an implementation of the same Action and associations between the Object's instance of that verb and the original verb.

At the association level when met a term like "action over an object" ("Increase **P**rice") the Agent starts to search a link between the "action" and its corresponding instance within the "object"'s action set. The presence of such a link means that the given action is applicable to that object so the Agent will implement that term; otherwise the term is treated as nonsense.

Thus, for example, "Increase" action will work differently for P and Q (assuming that it is implemented for both of them).

3.4. Defining Situation of TAC SCM

We do not consider the TAC market model because it is implemented on the server side, and according to the rules is inaccessible, so we have to deal only on the TAC agent's side. That is why we will define a TAC situation by including only the states of agent-related parameters. All the agent's entities with their current conditions and values compose a situation for the TAC agent at a specific period of time. It includes: factory utilization, warehouse of supplies with quantities of any present components, bank account, orders from customers, orders to suppliers, current date, start date, etc. So there is not any entity associated to a situation, it is represented as parameters set from other entities.

Having defined the concept of situation, we can use it when forming a knowledge base. The knowledge base, in turn, consists of a set of knowledge elements. And a knowledge element, in general, has the following structure: an initial situation, the applied action (or SP) and the resulting situation.

Moves (or strategy plans) in action may consider situations directly whenever they include any conditional content.

3.5. Embedding TAC SCM Interface into RGT Solver

The TAC Agent template has its own interface; we just duplicate its data monitors into our interface to track the game progress. For that we integrate a data transfer module into the RGT Solver and the TAC Agent's template in order to transfer the ongoing data between each other.

The TAC Solver module handles all the objects defined for the TAC by the Solver interface (namely: entities, actions, moves, strategies), so it is actually considered as the Agent's "brain". The module may be integrated either in the Agent or in the RGT Solver.

On the first case the Agent will transfer only the information to be displayed on the Solver's TAC gameboard tab. On the second case at every game event the Agent will pass the incoming information from the server to the Solver to be handled. After that the Solver processes the data and passes the results back to the Agent for further sending to the TAC server.

Conclusion

The RGT Solver framework with two integrated management-focused problems (the ValueWar strategy assessment tool and the TAC SCM game) is represented.

The ways for constructing objects, actions and strategies for the given management problems are described. Particularly:

- the ways for construction of operating entities, actions and strategy plans for ValueWar tool;
- the ways for construction of operating entities, actions, moves and strategy plans for TAC SCM game.

Also, we embedded the corresponding interfaces of each ValueWar and TAC SCM problems into separate tabs within the main Solver framework.

References

[1] M. Chussil and D. Reibstein, "Putting the lessons before the test. In Wharton on to Analyse & Develop Competitive Strategies", *John Wesley & Sons*, pp 343-368, 1997

[2] M. Chussil, D. Reibstein, *Strategy Analysis with Value War*, The SciPress, 1994.

[3] E. Pogossian, "Focusing management strategy provision simulation", *CSIT2001 3d International Conference in Computer Science and Information Technologies*, 5p., 2001.

[4] TAC SCM web page: *http://tac.sics.se/page.php?id=1*

[5] T. Baghdasaryan, E. Danielyan and E. Pogossian, "Testing oligopoly strategy plans by their on the job performance simulation", *International Conference CSIT2005*, 8p, 2005.

[6] T. Baghdasaryan, E. Danielyan and E. Pogossian, "Supply chain management strategy provision by game tree dynamic analysis", *Fifth International Conference SBM2006*, 2p., 2006.

[7] T. Baghdasaryan, "Scripting language and design tool for trading agents strategy plans", *CSIT2007 6th International Conference in Computer Science and Information Technologies*, 5p., 2007.

[8] T. Baghdasaryan, A. Grigoryan and Z. Naghashyan, "Development of a scripting language interpreter for acquisition of expert knowledge in a regular way", *CSIT2009 7th International Conference in Computer Science and Information Technologies*, 5p., 2009.

[9] Z Naghashyan and E. Pogossian, "Developing Java software for representation, acquisition and management of strategy knowledge", *Mathematical Problems of Computer Sciences*, pp.187-195, 2010.

[10] K. Khachatryan and V. Vagradyan, "Graphical language interpreter unified for SSRGT problems and relevant complex knowledge", *CSIT2011 8th International Conference in Computer Science and Information Technologies*, pp. 178-182, 2011.

[11] E. Pogossian, "Effectiveness enhancing knowledge based strategies for SSRGT class of defense problems", *Proceedings of NATO Advanced Study Institute (ASI)*, 2011.

[12] K. Khachatyan and S. Grigoryan “Java programs for presentation and acquisition of meanings in SSRGT games”, *Proceedings of SEUA Annual conference*, 7p., 2012.
[13] K. Khachatryan and S. Grigoryan, “Java programs for matching situations to the meanings of SSRGT games”, *Proceedings of SEUA Annual conference*, 5p., 2012

Submitted 30.11.2012, accepted 02.02.2013.

RGT Solver համակարգի ինտերֆեյսի համակերպումը կառավարման ռազմավարությունների փնտրման խնդիրներին

Թ. Բաղդասարյան

Ամփոփում

Մշակվում է RGT Solver ունիֆիկացված ծրագրային համակարգ այն դասի խնդիրների լուծման համար, որում հնարավոր լուծումների տիրույթը ներկայացվում է վերարտադրվող խաղային ծառերի միջոցով:

Նկարագրվում է RGT Solver համակարգի ինտերֆեյսի համակերպումը տվյալ դասին պատկանող մարքեթինգի և մատակարարման շղթայի կառավարման խնդիրներին, որոնք ներկայացված են համապատասխանաբար մարքեթինգի ռազմավարությունների վերլուծման ValueWar գործիքային փաթեթով և մրցակցող առևտրային գործակալներ (TAC SCM) խաղով:

Մասնավորապես, նկարագրվում է հետևյալը.
- ValueWar թաղանթի համար բաղադրիչների, գործողությունների և ռազմավարական պլանների նկարագրման եղանակները,
- TAC SCM մոդելի համար բաղադրիչների, գործողությունների, քայլերի և ռազմավարական պլանների նկարագրման եղանակները:

Адаптация решателя RGT-игр к задачам поиска стратегий менеджмента

Т. Багдасарян

Аннотация

Мы разрабатываем унифицированный механизм (RGT Solver) для решения класса задач, в которых пространство возможных решений представляется воспроизводимыми игровыми деревьями (RGT).

В рамках статьи мы описываем адаптацию интерфейса RGT Solver к задачам маркетинга и управления цепью снабжения, относящихся к данному классу и представленными, соответственно, моделями ValueWar и конкурирующего торгового агента (TAC SCM).

В частности, мы описываем:
- способы описания компонент, действий и стратегических планов для оболочки ValueWar;
- способы описания компонент, действий, ходов и стратегических планов для модели TAC SCM.